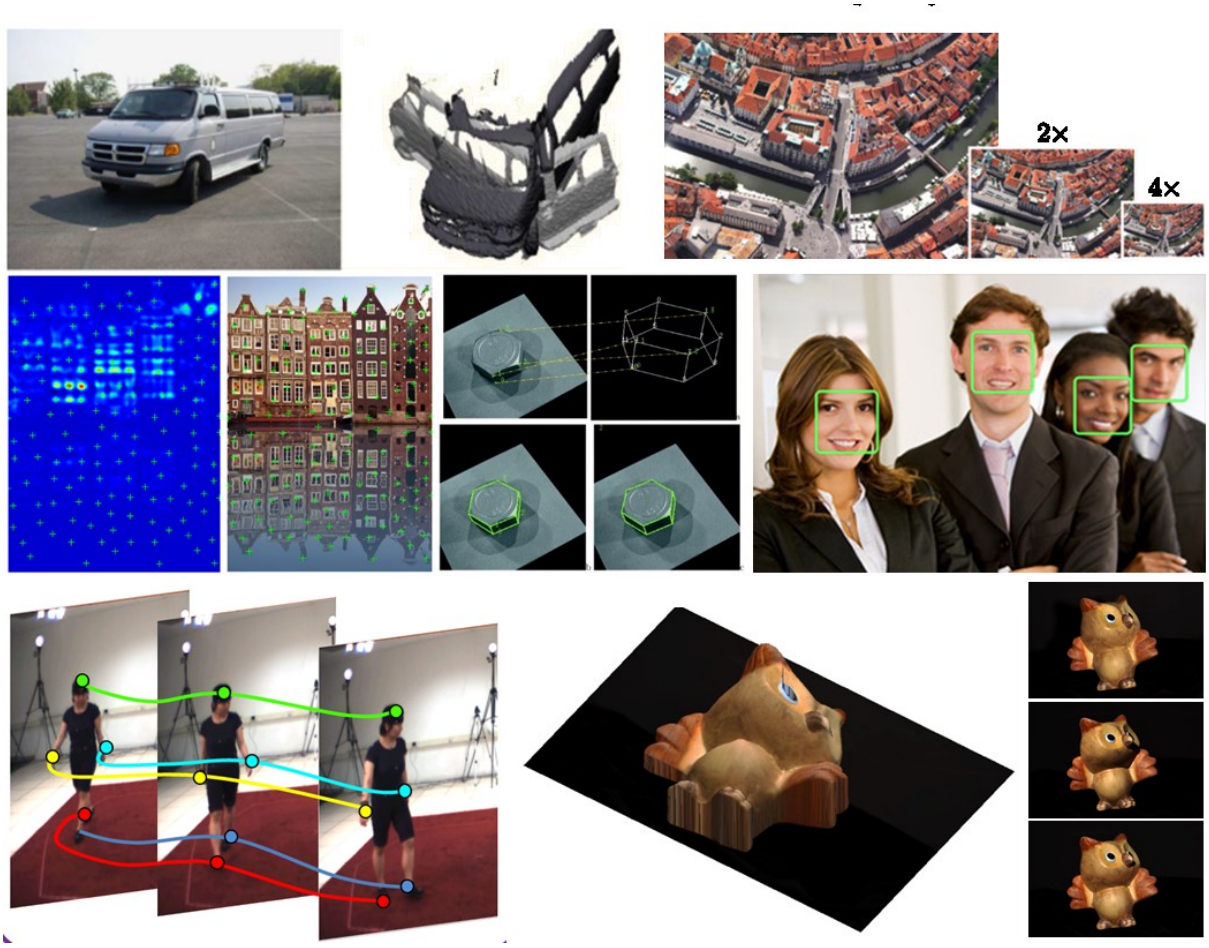
Univerza v Ljubljani
Fakulteta za elektrotehniko

Žiga Špiclin

Robotski vid

Laboratorijske vaje v programskem jeziku Matlab



Univerza v Ljubljani
Fakulteta za elektrotehniko

Žiga Špiclin

Robotski vid

Laboratorijske vaje v programskem jeziku Matlab

Ljubljana, 2016

Predgovor

Pričujoča zbirka nalog predstavlja dopolnilno študijsko gradivo pri predmetu Robotski na Univerzitetnem študiju elektrotehnike 2. stopnje, smer Robotika. Nastala je iz navodil za izvedbo laboratorijskih vaj pri tem predmetu v študijskem letu 2015/2016.

Namen gradiva je seznaniti študente z navodili laboratorijskih vaj in podati smernice za njihovo izvedbo. Zbirka nalog obsega 11 vaj, ki študente seznanijo z uporabo Matlaba za namen manipulacije, obdelave in analize digitalnih slik, postopki sivinskih in prostorskih preslikav slik, postopki glajenja in ostrenja slik ter njihovo interpolacijo in decimacijo, z algoritmi zaznave oglišč, robov in parametričnih struktur slike, z zaznavo predlog, geometrijsko kalibracijo slikovnega sistema, tehniko fotometrični stereo za rekonstrukcijo 3D oblik, postopki poravnave 3D modelov na podlagi 2D slik in osnovnimi principi odločanja na podlagi značilnic izločenih iz slik.

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

Ljubljana, November 2016 Žiga Špiclin

Kazalo

Vaja 0: Uvod v Matlab

Navodila in naloge

Uvodna vaja služi spoznavanju osnovnih ukazov za nalaganje, prikazovanje shranjevanje slik in upravljanje z digitalnimi slikami v okolju Matlab.

1. Datoteka *slika-8bit.raw* vsebuje dvodimenzionalno (2D) sliko v obliki surovih podatkov (RAW). Slika velikosti X x Y = 975 x 650 slikovnih elementov je zapisana z 8 biti na slikovni element. Napišite funkcijo za nalaganje poljubnih RAW slik:

   ```
   function oImage = loadImage( iPath, iSize, iFormat ),
   ```

   kjer je `iPath` pot do slike (mapa in ime datoteke), `iSize` vektor velikosti slike (v slikovnih elementih), `iFormat` pa oblika zapisa (tip podatka). Funkcija naj vrne naloženo sliko `oImage` v obliki matrike. Uporabite funkcijo `fread()` v povezavi s funkcijama `fopen()` in `fclose()`.

2. Prikažite sliko na zaslon z uporabo funkcije `image()`. Prikaz sivin prilagodite zapisu s pomočjo funkcije `colormap()`. Razmerje osi prikaza prilagodite razmerju osi slike z ukazom `axis image`. Napišite funkcijo za prikazovanje poljubnih RAW slik:

   ```
   function showImage( iImage, iTitle ),
   ```

   kjer je `iImage` slika, ki jo želite prikazati, `iTitle` pa naslov prikaznega okna. Slika naj se prikaže v novem oknu. Pomagajte si s funkcijama `figure()` in `title()`.

3. Napišite funkcijo za shranjevanje RAW slik:

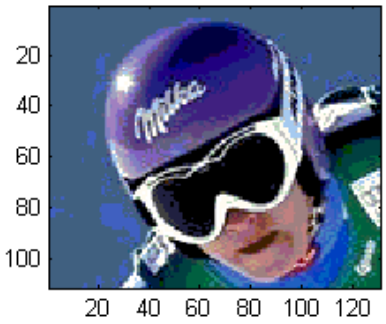
   ```
   function saveImage( iImage, iPath, iFormat ),
   ```

   kjer je `iImage` slika, ki jo želite shraniti, `iPath` pot do slike (mapa in ime datoteke), `iFormat` pa oblika zapisa. Uporabite funkcijo `fwrite()` v povezavi s funkcijama `fopen()` in `fclose()`.

4. Datoteka *slika-16bit.raw* vsebuje 2D sliko v obliki surovih podatkov (RAW), vendar pa je slika velikosti X x Y = 975 x 650 slikovnih elementov zapisana s 16 biti na slikovni element. Preizkusite delovanje zgoraj navedenih funkcij `loadImage()`, `showImage()` in `saveImage()`.

5. Datoteka slika-24bit-rgb.raw vsebuje barvno 2D sliko v obliki surovih podatkov (RAW), slika pa ima velikost X x Y = 975 x 650 slikovnih elementov, posamezen slikovni element pa je predstavljen s 24=3x8 biti. Barvna RAW slika je zapisana kot zaporedje treh 8 bitnih slik, kjer si sledijo rdeča, zelena in modra komponenta. Dopolnite funkcije `loadImage()`, `showImage()` in `saveImage()`, tako da bodo omogočale nalaganje, prikaz in shranjevanje barvnih RGB slik.

6. Iz barvne 2D slike izluščite pravokotno področje med oglišči (x_1,y_1)= (260,70) in (x_2,y_2)=(390,180) in ga prikažite na zaslon. Preverite pravilnost določanja področja s pomočjo spodnje slike.



7. Zamenjajte vrednosti slikovnih elementov rdečega kanala 2D barvne slike, ki imajo vrednost med 160 in 200 z vrednostjo 255. Uporabite funkcijo `find()`. Prikažite novo barvno sliko in jo primerjajte z originalno barvno sliko.

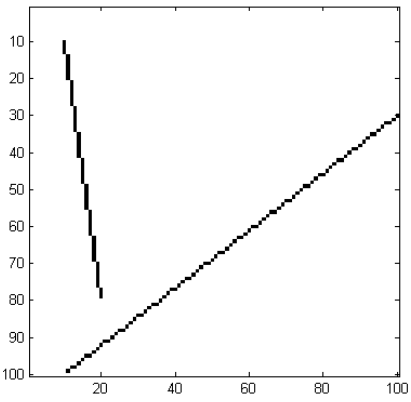
8. Pretvorite barvno sliko v sivinsko sliko in narišite projekciji maksimalne in povprečne svetlosti v smeri x in y smeri 2D slik. . Pomagajte si s funkcijami `max()`, `mean()` in `plot()`.

9. S pomočjo psevdo-algoritma, ki je podan spodaj, napišite funkcijo za risanje daljic v sliko:

`function oImage = drawLine( iImage, iValue, x1, y1, x2, y2 ),`

kjer je `iImage` slika, v katero želite risati daljico, `iValue` vrednost, ki jo pripišete slikovnemu elementu na daljici, `x1, y1` in `x2, y2` pa so koordinate začetne in končne točke daljice. Preverite delovanje funkcije, tako da ustvarite sliko velikosti X x Y = 100 x 100 in vse elemente postavite na vrednost 255, nato vrišite dve daljici s sivinskimi vrednostmi 0 in krajiščema (x_1,y_1)= (10,10) in (x_2,y_2)=(20,80) ter (x_1,y_1)= (100,30) in (x_2,y_2)=(10,100). Dobljeno sliko primerjajte s spodnjo sliko.

```
funkcija narišiDaljico(slika, vrednost,
x1, y1, x2, y2)
   dx := abs(x2-x1)
   dy := abs(y2-y1)
   če x1 < x2 potem sx := 1 sicer sx := -1
   če y1 < y2 potem sy := 1 sicer sy := -1
   napaka := dx-dy
zanka
     slika(x0,y0) = vrednost
     če x0 = x1 in y0 = y1 potem
       izhod iz zanke
     e2 := 2*napaka
     če e2 > -dy potem
       napaka := napaka - dy
       x0 := x0 + sx
     če e2 <  dx potem
       napaka := napaka + dx
       y0 := y0 + sy
konec zanke
```



10. Spremenite funkcijo za risanje daljic `drawLine()` tako, da bo omogočala risanje daljic z izbrano barvo. Ustvarite 2D barvno sliko velikosti X x Y = 300 x 100 in vse slikovne elemente obarvajte belo. S pomočjo posodobljene funkcije za risanje barvnih daljic v 2D sliko izpišite vaše ime, pri čemer ročno določite ustrezna krajišča daljic. Vsaka črka vašega imena naj ima drugo barvo.

Vprašanja

1. Kako bi morali spremeniti funkcijo za nalaganje slike, če veste da je slika zapisana v formatu vrstice-stolpci (in ne v formatu stolpci vrstice, kot je običajno v Matlabu)?
2. Kakšno vlogo ima pri prikazovanju slik Matlabov ukaz `axis image`?
3. Shranite 8-bitno sliko v formatu `uint32` (`single`, `double`) v RAW zapisu. Primerjajte velikost shranjene datoteke z originalno datoteko. Zakaj imata datoteki različni velikost?
4. Kakšno je razmerje velikosti med barvno RGB sliko in enako, a 8-bitno sivinsko sliko, če sta obe sliki zapisani v RAW formatu?
5. Določite povprečno svetlost 8-bitne (16-bitne) sivinske slike.
6. Prikažite desno spodnjo četrtino slike.
7. Kako bi izračunali središčno točko v sliki?
8. Iz barvne RGB slike izluščite kvadratno območje velikost 50 x 50 elementov s centrom v središčni točki in ga prikažite.
9. Iz desnega zgornjega kota barvne RGB (ali sivinske) slike izluščite pravokotno območje velikost 100 x 50 elementov in ga prikažite?
10. V barvni RGB sliki zamenjajte vrednosti slikovnih elementov zelenega kanala, ki so manjše od 100 z vrednostjo 0 in prikažite sliko?
11. Skicirajte projekcijo maksimalne in povprečne svetlosti sivinske slike vzdolž x smeri slike za like daljica, krog, trikotnik, pravokotnik? Za primer daljice preverite delovanje.
12. Ustvarite sliko velikosti X x Y = 100 x 100, vrišite premico $10x + 5y - 30 = 0$ in prikažite sliko.

Dodatne naloge

1. Daljica, ki nariše funkcija `drawLine()` zrnata. Razmislite, na kakšen način bi odpravili zrnatost daljice, tako da bi dobili gladko daljico? Namig: slikovnim elementom, ki predstavljajo daljico pripišite sivinske vrednosti v odvisnosti od njihove oddaljenosti od daljice. Slikovni elementi, ki pripadajo daljici naj bodo tisti, ki so od daljice oddaljeni do največ 1 slikovni element.
2. Postopek za risanje daljic, takoimenovani Bresenhamov algoritem, lahko nadgradimo za risanje digitalnih krogov in elips. Preverite izpeljavo postopka in napišite Matlab funkcijo, ki bo omogočala risanje elips.
3. Spremenite in uporabite funkcijo za risanje daljic tako, da bo omogočala izračun poljubnih poševnih projekcij sivinskih 2D slik. Pri tem namesto vpisovanja podatkov v sliko podatke preberite iz slike in uporabite za izračun željene projekcije. Izrišite projekciji maksimalne svetlosti za kota ϕ_1=30 in ϕ_2 = 135 stopinj glede na x smer 2D slike.

Vaja 1: Parametri kakovosti slik

Navodila in naloge

Vaja je namenjena spoznavanju in razumevanju osnovnih lastnosti realnih slik kot so svetlost, dinamično območje, histogram, kvantizacija, šum in barva ter spoznavanju načinov za določanje značilnih parametrov kakovosti realnih slik kot je razmerje signal-šum.

1. S pomočjo fotoaparata zajemite dve sliki kalibra ColorChecker, prvo ko so v predavalnici prižgane luči (*slika A*) in drugo ko so luči ugasnjene (*slika B*). Bliskavica naj bo pri tem izklopljena. Sliki prenesite na računalnik in jih uvozite in prikažite v Matlabu s funkcijama `imread()` in `imshow()`. Iz zajetih slik izluščite pravokotna področja oziroma podslike tako, da bo vsako področje vsebovalo le eno barvo oziroma sivino. Podslike shranite v vrstični vektor tipa `cell`, v enakem vrstnem redu kot je označen na spodnji sliki. Za hitro in enostavno določanje pravokotnih področij na sliki lahko označite le položaje nasprotnih si oglišč pravokotnika, tako da uporabite Matlabovo funkcijo `ginput()`.

2. Posamezne podslike zajetih slik A in B pretvorite v sivinski sliki po enačbi $S = 0{,}299R + 0{,}587G + 0{,}114B$. Za vsako podsliko izračunajte povprečne 8-bitne sivinske vrednosti in poiščite indekse področij v slikah A in B, ki pripadajo poslikam z najmanjšo in največjo povprečno 8-bitno sivinsko vrednostjo.

   a. Razmislite, katero področje bi moralo imeti najmanjšo in katero največjo povprečno 8-bitno sivinsko vrednost. Navedite pripadajoče indekse območij.

   b. Sliki A in B sta bili zajeti pod različnima osvetlitvama. Ali osvetlitev vpliva na to, katero področje v slikah A in B ima najmanjšo in katero največjo povprečno 8-bitno sivinsko vrednost?

3. Narišite graf, ki prikazuje povprečne sivinske vrednosti področij *19-24* v odvisnosti od pripadajočega indeksa območja.

   a. Kakšna je oblika grafa in kako se razlikujeta grafa slik A in B?

4. Histogram slike je grafično orodje za prikazovanje frekvenčne porazdelitve sivinskih vrednosti v slikovnih elementih na sliki. Abscisna os histograma predstavlja območje sivinskih vrednosti, ordinatna os histograma pa podaja število slikovnih elementov v sliki z izbrano sivinsko vrednostjo. Napišite funkcijo za prikaz histograma slike:

```
function oHist = displayHistogram(iImage, iNumBins, iTitle),
```

kjer je `iImage` slika, `iNumBins` število podintervalov območja sivinskih vrednosti, `iTitle` pa naslov prikaznega okna. Funkcija vrne histogram slike v obliki vrstičnega vektorja dolžine `iNumBins`. Histogram izračunajte brez uporabe Matlabovih naprednih funkcij, za prikazovanje pa uporabite funkcijo `bar()`.

a. Kakšen vpliv ima število podintervalov `iNumBins` na izgled histograma slike?

b. Prikažite histogram sivinske slike A. Čemu v sliki A ustrezajo posamezni vrhovi v histogramu?

c. Prikažite in primerjajte histograma sivinskih slik A in B. V čem se histograma slik A in B bistveno razlikujeta?

5. Napišite funkcijo, ki preslika trenutni razpon sivinskih vrednosti vhodne 8-bitne sivinske slike `iImage` tako, da izkoristi celotni razpon 8-bitnih števil [0, 255]. Funkcija naj vrne obdelano sliko `oImage`:

```
function oImage = equalizeImage( iImage ).
```

Funkcijo uporabite na 8-bitnih sivinski sliki A.

a. Ali se oblika histograma slik A in B spremeni po uporabi funkcije `equalizeImage()`?

6. Dopolnite funkcijo `equalizeImage()` tako, da bo omogočala preslikavo trenutnega razpona sivinskih vrednosti vhodne 8-bitne sivinske slike `iImage` na poljuben razpon celoštevilskih vrednosti. Funkcijo preizkusite s preslikavo 8-bitne sivinske slike A na 4- in 16-bitni razpon. Bodite pozorni, da bo funkcija izhodno sliko pretvorila v ustrezen podatkovni tip, pri tem pa si pomagajte z Matlabovima funkcijama `class()` in `cast()`.

a. Kako se spremeni število med seboj različnih povprečnih sivinskih vrednosti na ustreznih podslikah originalne 8-bitne in preslikane 4-bitne sivinske slike A?

b. Ali se število med seboj različnih številskih vrednosti kakorkoli spremeni pri preslikavi iz 8-bitne v 4- oziroma 16-biten podatkovni tip?

7. Šum je izraz za kakršnokoli naključno spreminjanje sivinskih vrednosti in ima lahko pomemben vpliv na kakovost slike. Izvor, vrsta in velikost šuma so odvisni od fizikalnega procesa oziroma načina zajemanja slik, najbolj znani in splošno uporabni model šuma pa je Gaussov šum. Gaussov šum je definiran s funkcijo gostote verjetnosti $p(z) = {}^{1}/_{\sqrt{2\pi}\sigma}\, e^{-0{,}5\,(z-\mu)^2/\sigma^2}$, kjer μ predstavlja srednjo vrednost, σ pa standardno deviacijo šuma. Šum v sliki ocenjujemo na področjih s predvidoma konstantno sivinsko vrednostjo, tako da sivinske vrednosti preslikamo v histogram in ga normaliziramo. Normaliziran histogram predstavlja oceno relativne verjetnostne porazdelitve $p_r(z)$ s katero določimo parametra μ in σ Gaussove porazdelitve kot $\mu = \sum_{i=1}^{L} z_i\, p_r(z_i)$ in $\sigma = \sqrt{\sum_{i=1}^{L} (z_i - \mu)^2\, p_r(z_i)}$.

a. Preverite ali je Gaussov šum ustrezen model šuma za vaše slike. Odgovor ustrezno utemeljite.

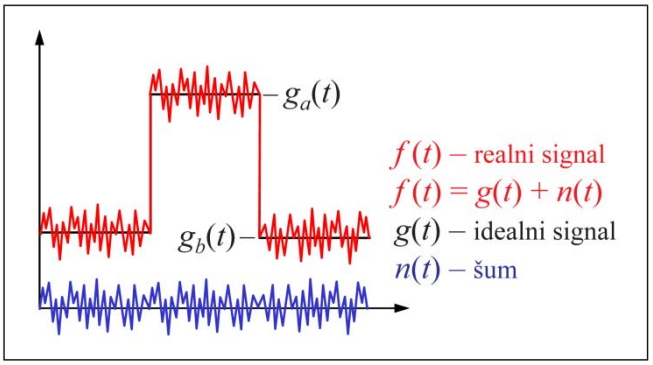
b. Za vsako podsliko slik A in B izračunajte parametra Gaussovega šuma μ in σ ter izrišite na zaslon pripadajoča razsevna diagrama $\sigma(\mu)$. V čem se razlikujeta razsevna diagrama slik A in B?

8. Razmerje signal-šum (SNR) je pomemben kriterij za vrednotenje relativne jakosti oziroma moči signala glede na prisoten šum. SNR je namreč merilo zanesljivosti oziroma sposobnosti zaznavanja prisotnosti sprememb v opazovanem signalu. Uporabljajo se trije osnovni načini podajanja SNR, in sicer amplitudni (SNR_A), diferencialni (SNR_D) ter močnostni (SNR_M) način. Vendar pa podajanje SNR ni standardizirano , ampak je odvisno od vrste signala in šume ter od namena uporabe. V primeru dveh nivojev signala je SNR_D podan kot:

$$SNR_D = \frac{|\mu_A - \mu_B|}{\sqrt{{\sigma_A}^2 + {\sigma_B}^2}}$$

a. Izračunajte diferencialno razmerje signal-šum (SNR_D) med podslikama z indeksom *19* in *24* v sivinskih slikah A in B. Katera slika ima večji SNR_D in zakaj?

b. Izračunajte dvojiški logaritem diferencialnega razmerja signal-šum (SNR_D). Kaj vam pove ta vrednost?



9. Barvo slikovnega elementa običajno definiramo s tremi, včasih pa celo z dvema komponentama oziroma vrednostima. Zaradi načina pretvorbe svetlobe v digitalni zapis se najpogosteje uporablja zapis barve slikovnega elementa s komponentami RGB, ki ustrezajo trem različnim detektorjem svetlobe in ki so selektivno občutljivi pri valovnih dolžinah svetlobe okoli 700 nm (R), 550 nm (G) in 450 nm (B). Obstajajo tudi drugi barvni prostori, ki so bolj primerni za analizo digitalnih slik, naprimer XYZ in L*a*b*. Sliko v RGB barvnem prostoru lahko pretvorimo v drug barvni prostor z (ne)linearno preslikavo RGB komponent. Preslikava iz RGB v XYZ barvni prostor pri referenčni beli osvetlitvi D65 je definirana kot:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0{,}4124564 & 0{,}3575761 & 0{,}1804375 \\ 0{,}2126729 & 0{,}7151522 & 0{,}0721750 \\ 0{,}0193339 & 0{,}1191920 & 0{,}9503041 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix},$$

pri čemer morajo vrednosti komponent R, G in B ležati na intervalu [0, 1]. Preslikava iz XYZ v L*a*b* barvni prostor je definirana kot:

$$\begin{aligned} L^* &= 116\, Y^{1/3} - 16, \\ a^* &= 500\, (X^{1/3} - Y^{1/3}), \\ b^* &= 200\, (Y^{1/3} - Z^{1/3}). \end{aligned}$$

L*a*b* barvni prostor je zaradi linearne metrike med različnimi barvami še posebej primeren za kvantitativno primerjavo barv in barvnih odtenkov.

a. Pretvorite sliko A iz RGB v XYZ barvni prostor in prikažite posamezne X, Y in Z komponente kot sivinske slike. Kvalitativno ocenite na katerih podslikah je prišlo do največjih sprememb?

b. Pretvorite sliko A, ki je zapisana z XYZ komponentami, v L*a*b* barvni prostor in prikažite L*, a* in b* komponente slik kot sivinske slike. Katere podslike L*a*b* se bistveno spremenijo pri tej pretvorbi?

c. Izračunajte Evklidsko razdaljo med povprečnimi vrednostmi RGB komponent slike A v podsliki z indeksom 19 in podslikami z indeksi 1-18. Ustrezne vrednosti izračunajte tudi za L*a*b* komponente slike A. V katerem polju pride do največjih razlik med merama?

Vaja 2: Sivinske preslikave slik

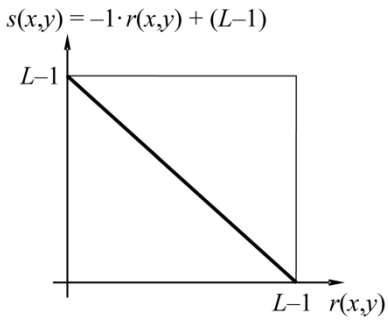
Navodila

Sivinske preslikave so poljubne preslikave $T: R \to R$ dinamičnega območja $[0 \dots L_{r-1}]$ referenčne slike $r(x,y)$ v dinamično območje $[0 \dots L_{s-1}]$ preslikane slike $s(x,y)$. Preslikavo izvedemo na vseh slikovnih elementih slike kot $s(x,y) = T(\, r(x,y)\,)$. Glavni namen sivinskih preslikav je **povečevanje kontrasta** struktur zanimanja, uporabljajo pa se tudi za **prilagoditev dinamičnega območja** referenčne slike za potrebe prikazovanja.

- **Linearno preslikavo** izvedemo tako, da sivinsko vrednost vsakega slikovnega elementa $r(x,y)$ pomnožimo s konstanto a (sprememba kontrasta slike) in zmnožku prištejemo konstanto b (sprememba svetlosti slike):

  $$s(x,y) = a \cdot r(x,y) + b.$$

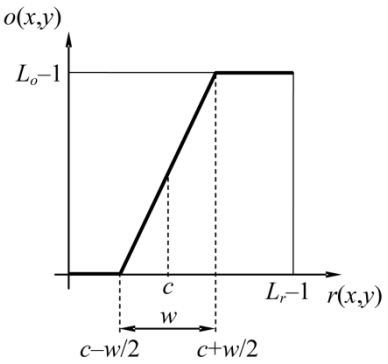
  Rezultat je slika $s(x,y)$, ki ima linearno preslikave sivinske vrednosti.



- **Linearno oknenje** izvedemo tako, da na dinamičnem območju referenčne slike definiramo poljubno okno s središčem c in širino w. Sivinske vrednosti v sliki, ki so manjše od $c - {}^{w}/_{2}$ preslikamo v vrednost 0; tiste, ki so večje od $c + {}^{w}/_{2}$ pa v vrednost L_{o-1}; sivinske vrednosti na intervalu od $c - {}^{w}/_{2}$ do $c + {}^{w}/_{2}$ preslikamo z linearno preslikavo po enačbi:

  $$o(x,y) = \frac{L_o - 1}{w} \cdot \left(r(x,y) - \left(c - \frac{w}{2} \right) \right).$$

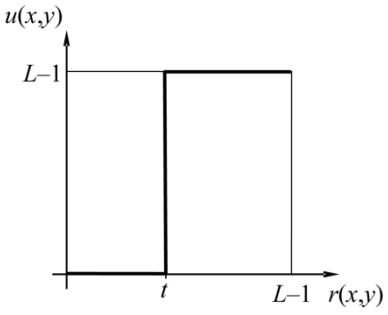
  Rezultat je slika $o(x,y)$, ki ima sivinske vrednosti znotraj okna linearno razširjene na celotno dinamično območje $[0 \dots L_{o-1}]$.



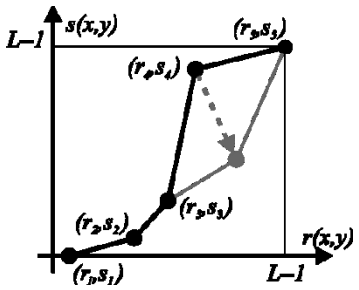
- **Upragovljanje** izvedemo tako, da sivinsko vrednost vsakega slikovnega elementa referenčne slike $r(x,y)$, ki je manjša ali enaka izbrani vrednosti praga t, preslikamo v vrednost 0, sicer pa v največjo možno sivinsko vrednost $L - 1$:

  $$u(x,y) = \begin{cases} 0 & \text{pri } r(x,y) \le t. \\ L-1 & \text{pri } r(x,y) > t. \end{cases}$$

  Rezultat je upragovljena slika $u(x,y)$.



- **Preslikava po odsekih** je popolnoma določena z množico urejenih parov kontrolnih točk $(r_i, s_i); i = 1 \dots N$. Zaporedni pari kontrolnih točk določajo linearno preslikavo iz dinamičnega območja $[r_i, r_{i+1}]$ referenčne slike v dinamično območje preslikane slike $[s_i, s_{i+1}]$ po enačbi:

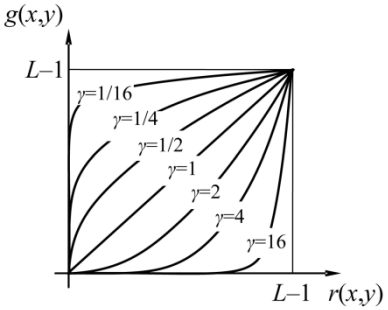


$$s(x,y) = \frac{s_{i+1} - s_i}{r_{i+1} - r_i}\,(r(x,y) - r_i) + s_i$$

Na odsekih, kjer so multiplikativni koeficienti linearne preslikave večji od 1, kontrast povečujemo, sicer pa ga zmanjšujemo.

- **Gama preslikava** je zvezna nelinearna preslikava, ki je v primeru enakega dinamičnega območja referenčne in preslikane slike $L_r = L_o = L$ definirana kot:

$$g(x,y) = (L-1)^{1-\gamma} r^{\gamma}(x,y).$$



Rezultat je gama preslikana slika $g(x,y)$. Gama preslikava je uporabna takrat, kadar želimo zvezno in nelinearno povečati kontrast svetlejših področij na račun manjšega kontrasta temnejših področij ($\gamma > 1$), oziroma obratno ($\gamma < 1$).

Naloge

Pri vaji boste napisali funkcije sivinskih preslikav in preizkusili njihovo delovanje na sivinski in barvni sliki.

1. Naložite barvno RGB sliko *slika.jpg* v okolju Matlab in jo pretvorite v sivinsko sliko po enačbi $S = 0{,}299R + 0{,}587G + 0{,}114B$. Poskrbite za ustrezno pretvorbo in zaokroževanje sivinskih vrednosti slikovnih elementov z Matlabovima ukazoma `cast()` in `round()`, za nalaganje slike pa uporabite ukaz `imread()`.
2. Napišite funkcijo za poljubno linearno sivinsko preslikavo:

```
function oImage = scaleImage( iImage, iSlope, iIntersection),
```

kjer je `iImage` slika, ki ji preslikujete sivinske vrednosti. `iSlope` in `iIntersection` pa sta parametra linearne preslikave (parametra premice a in b). Funkcija vrne linearno preslikano sivinsko sliko `oImage`. Poskrbite, da bo imela izhodna slika enak podatkovni tip kot vhodna slika.

   a. Preizkusite delovanje funkcije na sivinski sliki za različne vrednosti `iSlope` ter `iIntersection`. Narišite histogram poljubno preslikane sivinske slike in obrazložite, kako poljubna linearna sivinska preslikava vpliva na histogram slike.

   b. Določite vrednosti `iSlope` ter `iIntersection` linearne sivinske preslikave tako, da se bo najsvetlejša točka slike preslikala v najtemnejšo točko slike in nasprotno.

   c. Razširite funkcijo `scaleImage()` tako, da bo omogočala linearno preslikovanje barvnih slik in preizkusite delovanje z različnimi vrednostmi `iSlope` ter `iIntersection`.

3. Napišite funkcijo za poljubno linearno sivinsko oknenje:

```
function oImage = windowImage( iImage, iCenter, iWidth ),
```

kjer je `iImage` slika, ki ji preslikujete sivinske vrednosti, `iCenter` in `iWidth` pa sta parametra linearnega oknenja (središče okna c in širina okna w). Funkcija vrne linearno sivinsko oknjeno sliko `oImage`.

   a. Preizkusite delovanje funkcije na sivinski sliki za vrednosti `iCenter` $= 64$ ter `iWidth` $= 192$.

   b. Določite vrednosti `iCenter` in `iWidth` tako, da boste z oknjenjem iz sivinske slike odstranili 10% najmanjših in 10% največjih sivinskih vrednosti. Pomagajte si z Matlabovo funkcijo `sort()`. Obrazložite vpliv rezultirajoče preslikave na originalno sivinsko sliko.

4. Napišite funkcijo za poljubno sivinsko upragovljanje:

```
function oImage = thresholdImage( iImage, iThreshold ),
```

kjer je `iImage` slika, ki ji preslikujete sivinske vrednosti, `iThreshold` pa je parameter upragovljanja (vrednost praga t). Funkcija vrne sivinsko upragovljeno sliko `oImage`.

   a. Preizkusite delovanje funkcije na sivinski sliki za vrednost `iThreshold` $= 127$.

5. Napišite funkcijo za poljubno preslikavo po odsekih:

```
function oImage =
multiscaleImage(iImage,iSrcPoints,iDstPoints),
```

kjer je `iImage` slika, ki ji preslikujete sivinske vrednosti, `iSrcPoints` je vrstični vektor $[r_1 \dots r_N]$ kontrolnih točk na dinamičnem območju vhodne slike `iImage`, `iDstPoints` je vrstični vektor $[s_1 \dots s_N]$ preslikanem kontrolnih točk, prav tako na dinamičnem območju vhodne slike `iImage`. Funkcija vrne po odsekih preslikano sivinsko sliko `oImage`.

   a. Preizkusite delovanje funkcije na sivinski sliki za dve skupini kontrolnih točk $r_1 = [0,127,255]$ in $s_1 = [0,191,255]$ ter $r_2 = [0,127,255]$ in $s_2 = [0,65,255]$. Obrazložite vpliv sivinske preslikave po odsekih na lastnosti sivinske slike in na lastnosti histograma pripadajočih, po odsekih preslikanih sivinskih slik.

6. Napišite funkcijo za poljubno gama sivinsko preslikavo:

```
function oImage = gammaImage( iImage, iGamma ),
```

kjer `iImage` slika, ki ji preslikujete sivinske vrednosti, `iGamma` pa je parameter gama (gama vrednost γ). Funkcija vrne gama sivinsko preslikano sliko `oImage`.

   a. Preizkusite delovanje funkcije na sivinski sliki pri poljubnih vrednostih `iGamma`. Obrazložite vpliv vrednosti `iGamma` na sivinske vrednosti in na histogram sivinske slike.

   b. Dopolnite funkcijo tako, da bo omogočala poljubno gama sivinsko preslikavo barvnih slik.

Vaja 3: Osnovna obdelava slik

Navodila

Vaja je namenjena spoznavanju in razumevanju osnovnih postopkov za obdelavo slik kot so filtriranje, glajenje in ostrenje ter interpolacija in decimacija slik.

Filtriranje slike lahko izvedemo s postopkom **2D diskretne konvolucije** med podano sliko $g(x, y)$ in konvolucijskim jedrom $k(u, v)$ velikosti $[m, n]$:

$$f(x, y) = \sum_{u=-m/2}^{m/2} \sum_{v=-n/2}^{n/2} k(u, v)\, g(x - u, y - v),$$

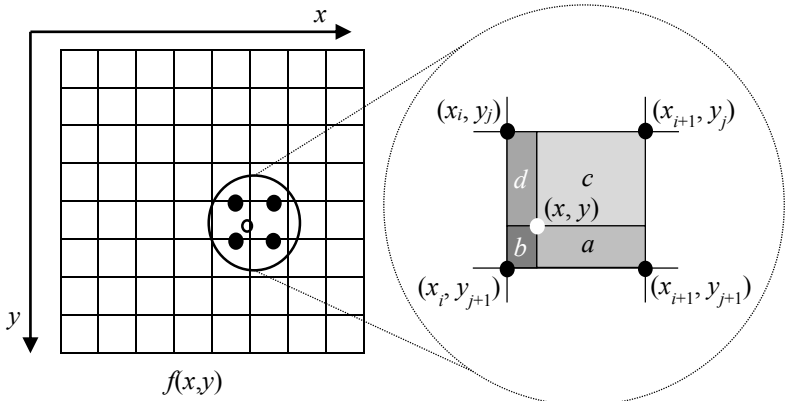
kjer je $f(x, y)$ izhodna slika. Na področju, kjer slika $g(x, y)$ ni definirana, v skladu z definicijo konvolucije predpostavimo sivinsko vrednost 0. V Matlabu je 2D konvolucijo mogoče izračunati s štirimi vgnezdenimi `for` zankami. Zunanji dve zanki uporabimo za naslavljanje slikovnih elementov slike $g(x, y)$, notranji dve zanki pa za naslavljanje konvolucijskega jedra $k(u, v)$. Konvolucijsko jedro $k(u, v)$ je definirano kot 2D matrika, središče jedra $k(0,0)$ pa v Matlabu ustreza indeksoma `floor((size(K)+1)/2)`, kar je potrebno upoštevati pri naslavljanju 2D matrike konvolucijskega jedra.

S postopkom **interpolacije slik** lahko priredimo sivinsko vrednost poljubni točki v slikovni ravnini. Na ta način lahko povečamo vzorčno frekvenco in s tem velikost slik ter tako navidezno zmanjšamo velikost slikovnih elementov. Glede na to, koliko sosednjih slikovnih elementov upoštevamo pri izračunu sivinske vrednosti v podani točki, delimo način interpolacije slik na:

7. **ničti red** ali interpolacija najbližjega soseda – upoštevamo le najbližji slikovni element,
8. **prvi red ali (bi)linearna interpolacija** – upoštevamo le štiri sosednje slikovne elemente,
9. **višji red**, npr. (bi)kubična interpolacija (drugi red), ki upošteva 16 sosednjih slikovnih elementov.

Računska zahtevnost interpolacijskih postopkov 2D slik v grobem raste s kvadratom reda interpolacije, kar pomeni, da je bikubična interpolacija (*drugi red*) približno štiri krat bolj zahtevna oziroma počasnejša od bilinearne (*prvi red*) interpolacije. Naštete interpolacijske postopke je mogoče posplošiti, tako da delujejo tudi za večrazsežne slike.

Bilinearna interpolacija

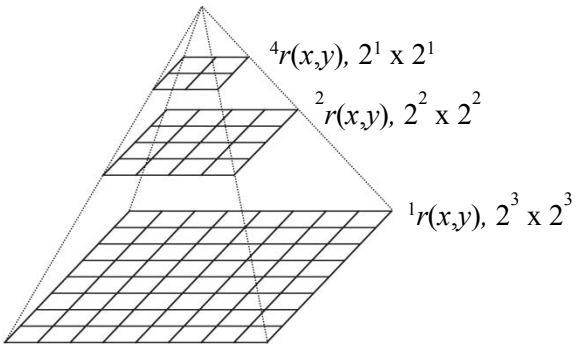


$$dx = (x_{i+1} - x_i)$$
$$dy = (y_{j+1} - y_i)$$
$$s = dx \cdot dy$$
$$a = (x_{i+1} - x) \cdot (y_{j+1} - y)$$
$$b = (x - x_i) \cdot (y_{i+1} - y)$$
$$c = (x_{i+1} - x_i) \cdot (y - y_j)$$
$$d = (x - x_i) \cdot (y - y_j)$$

$$f(x,y) = f(x_i, y_i) \cdot \frac{a}{s} + f(x_{i+1}, y_j) \cdot \frac{b}{s} + f(x_i, y_{j+1}) \cdot \frac{c}{s} + f(x_{i+1}, y_{j+1}) \cdot \frac{d}{s}$$

S postopkom **decimacije slik** zmanjšamo vzorčno frekvenco ter s tem velikost slik. Skladno z Nyquistovim vzorčnim teoremom je pred postopkom decimacije sliko potrebno filtrirati z nizko prepustnim sitom in na ta način odstraniti visoko frekvenčno informacijo. Pri decimaciji se pogosto uporablja piramidna shema, kjer se vzorčna frekvenca izvirne slike zaporedoma zmanjšuje s celoštevilsko vrednostjo, običajno dva.

Piramidna decimacijska shema



Primer dveh jeder nizko prepustnega sita

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{8}$
$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$

$\frac{1}{400}$	$\frac{1}{80}$	$\frac{1}{50}$	$\frac{1}{80}$	$\frac{1}{400}$
$\frac{1}{80}$	$\frac{1}{16}$	$\frac{1}{10}$	$\frac{1}{16}$	$\frac{1}{80}$
$\frac{1}{50}$	$\frac{1}{10}$	$\frac{4}{25}$	$\frac{1}{10}$	$\frac{1}{50}$
$\frac{1}{80}$	$\frac{1}{16}$	$\frac{1}{10}$	$\frac{1}{16}$	$\frac{1}{80}$
$\frac{1}{400}$	$\frac{1}{80}$	$\frac{1}{50}$	$\frac{1}{80}$	$\frac{1}{400}$

Naloge

Pri vaji boste napisali funkcije za filtriranje slik z 2D konvolucijo in preizkusili delovanje z različnimi jedri za namen glajenja in ostrenja sivinskih in barvnih slik. Funkcije za interpolacijo in decimacijo slik boste uporabili za povečavo oziroma pomanjšavo sivinskih in barvnih slik.

1. Napišite funkcijo, ki izračuna 2D diskretno konvolucijo med vhodno sivinsko sliko `iImage` in konvolucijskim jedrom `iKernel`:

   `function oImage = discreteConvolution2D( iImage, iKernel ),`

   ki vrne sivinsko sliko `oImage`.

   a. Naložite barvno RGB sliko *slika.jpg* v okolju Matlab in jo pretvorite v sivinsko sliko po enačbi $S = 0{,}299R + 0{,}587G + 0{,}114B$. Preizkusite delovanje funkcije 2D diskretne konvolucije na sivinski sliki s konvolucijskim jedrom `iKernel = ones(k)/k^2`.

Preizkusite delovanje funkcije za različne vrednost $k = 2n + 1, n = 1,2,3, \ldots$. Obrazložite vpliv 2D diskretne konvolucije z danim konvolucijskim jedrom na izhodno sliko. Katera operacija nad slikovnimi elementi v soseščini $k \times k$ je ekvivaletna danemu konvolucijskemu jedru?

Laplaceov operator

0	1	0
1	-4	1
0	1	0

b. Dopolnite funkcijo 2D diskretne konvolucije tako, da bo omogočala izračun konvolucije na barvnih slikah. Konvolucijo z barvno sliko izračunate na posameznem kanalu barvne slike in kanale združite v novo, izhodno barvno sliko `oImage`. Preizkusite delovanje funkcije 2D diskretne konvolucije na barvni sliki z uporabo konvolucijskega jedra `iKernel = ones(k)/k^2` različne vrednost $k = 2n + 1, n = 1,2,3, \ldots$.

2. Napišite funkcijo za izračun konvolucijskega jedra v obliki 2D simetrične Gaussove funkcije:

```
function oKernel = discreteGaussian2D( iSigma ),
```

kjer je `iSigma` standardna deviacija σ v 2D simetrični Gaussovi funkciji, ki je definirana kot:

$$k(u,v) = {}^{1}/_{2\pi\sigma^2}\, e^{-0{,}5\,(u^2+v^2)/\sigma^2}.$$

Velikost konvolucijskega jedra $k(u,v)$ nastavite glede na vrednost `iSigma` kot $2 \cdot \lceil 3 \cdot \sigma \rceil + 1$, vrednost $k(0,0)$ pa naj bo v središčnem elementu izhodne 2D matrike `oKernel`. Zagotovite, da bo vsota vseh elementov `oKernel` enaka 1.

a. Preizkusite delovanje funkcije za poljubne nenegativne vrednosti `iSigma` in prikažite konvolucijsko jedro z Matlabovo funkcijo `surf()`. Obrazložite vpliv vrednosti `iSigma` na obliko konvolucijskega jedra `oKernel`.

3. Napišite funkcijo za glajenje sivinskih in barvnih slik z 2D diskretno konvolucijo med vhodno sliko `iImage` in 2D simetrično Gaussovo funkcijo:

```
function oImage = imageSmoothing( iImage, iSigma ),
```

kjer je `iSigma` standardna deviacija 2D simetrične Gaussove funkcije, `oImage` pa izhodna sivinska slika.

a. Preizkusite delovanje funkcije na sivinski in barvni sliki za poljubne nenegativne vrednosti `iSigma` in prikažite izhodne sivinske in barvne slike. Obrazložite vpliv vrednosti `iSigma` na izhodne sivinske in barvne slike.

b. Kaj se po glajenju zgodi z zunanjim robom slike in zakaj? Kako širok je ta rob in od česa je odvisna širina tega roba? Kako bi se lahko izognili ali ublažili opažene spremembe.

c. Razširite funkcijo `discreteConvolution2D()` tako, da bo imela dodatni vhodni parameter `iPadImage`. Parameter `iPadImage` naj definira način razširitve slike pred izračunom diskretne 2D konvolucije. Naprimer, vhodno sliko `iImage` razširite z robom ustrezne širine, katerega sivinske vrednosti določa parameter `iPadImage ={'zeros', 'nearest'}`. Ko je vrednost parametra `iPadImage` enaka `'zeros'`, naj bo vrednost vseh sivin v razširjenem delu slike enaka 0. Ko je vrednost parametra `iPadImage` enaka `'nearest'`, naj bodo sivinske vrednosti v razširjenem delu slike enake najbližjim sivinam v originalni vhodni slike `iImage`. Funkcija `discreteConvolution2D()` naj vrne le tisti del konvolirane slike, ki pripada originalni vhodni sliki.

4. Ostrenje slike je analogno prostorskemu odvajanju sivinskih vrednosti. Najenostavnejši operator za drugi odvod je Laplaceov operator, ostrenje slik pa se izvede tako, da se vhodni sliki $g(x,y)$ odšteje uteženo sliko drugega odvoda slike:

$$f(x,y) = g(x,y) - c[\nabla^2 g(x,y)],$$

kjer konstanta c določa stopnjo ostrenja. Pogosto se uporablja tudi maskiranje neostrih področij, pri čemer vhodni sliki $g(x,y)$ najprej odštejemo njeno zglajeno verzijo $G(x,y)$ ter tako dobimo sliko maske $m(x,y)$, ki jo prištejemo vhodni sliki v skladu s stopnjo ostrenja c:

$$m\big(g(x,y)\big) = g(x,y) - G\big(g(x,y)\big) \xrightarrow{ostrenje}$$
$$f(x,y) = g(x,y) + c[m(g(x,y))].$$

Po odštevanju oz. prištevanju slike drugega odvoda $\nabla^2 g = (x,y)$ oz. slike maske $m(x,y)$ zagotovite, da bodo vrednosti sivinske slike na območju od [0, 255], pri čemer vrednosti manjše od 0 oz. vrednosti večje od 255 ustrezno zaokrožite.

a. Izostrite sivinsko sliko s postopkom na podlagi Laplaceovega operatorja in z maskiranjem neostrih področij. Za stopnjo ostrenja izberite vrednost $c = 2$. Preizkusite različne vrednosti c in obrazložite njihov vpliv na rezultirajočo sivinsko sliko.

5. Napišite funkcijo za interpolacijo ničtega reda, ki vzorči vhodno 2D sivinsko sliko `iImage`:

```
function oImage = interpolate0Image2D(iImage, iCoorX, iCoorY),
```

kjer so `iCoorX` in `iCoorY` vzorčne koordinate v prostoru vhodne slike `iImage`, katerih sivinsko vrednost moramo izračunati. Izračunane sivinske vrednosti predstavljajo sivinske vrednosti izhodne slike `oImage`. Interpolacija ničtega reda priredi sivinsko vrednost v točki (x,y) enako sivinski vrednosti najbližje točke na diskretni vzorčni mreži. Pri iskanju najbližje točke v diskretni mreži si pomagajte z Matlabovo funkcijo `round()`, za definicijo vzorčnih točk pa uporabite funkcijo `ndgrid()`.

a. Preizkusite delovanje funkcije tako, da prevzorčite sivinsko sliko s trikrat bolj gosto vzorčno mrežo, kot je mreža originalne sivinske slike. Kolikšna je velikost izhodne slike?

6. Napišite funkcijo za interpolacijo prvega reda, ki vzorči vhodno 2D sivinsko sliko `iImage`:

```
function oImage = interpolate1Image2D(iImage, iCoorX, iCoorY),
```

kjer so `iCoorX` in `iCoorY` vzorčne koordinate v prostoru vhodne slike `iImage`, katerih sivinsko vrednost moramo izračunati. Izračunane sivinske vrednosti predstavljajo sivinske vrednosti izhodne slike `oImage`. Interpolacija prvega reda priredi sivinsko vrednost v točki (x,y) enako linearno uteženi vsoti sivinskih vrednosti sosednjih štirih točk na diskretni vzorčni mreži (*glej opis bilinearne interpolacije*). Pri iskanju sosednjih točk v diskretni mreži si pomagajte z Matlabovo funkcijo `floor()`.

a. Preizkusite delovanje funkcije tako, da prevzorčite sivinsko sliko s trikrat bolj gosto vzorčno mrežo, kot je mreža originalne sivinske slike. Primerjajte dobljeno izhodno sliko z izhodno sliko po interpolaciji ničtega reda. Kaj so prednosti in slabosti interpolacije ničtega reda? Kaj so prednosti in slabosti interpolacije prvega reda?

b. Interpolacijo slike lahko uporabljamo za povečavo delov slike. S pomočjo funkcije za interpolacijo prvega reda izvedite petkratno (5x) povečavo pravokotnega področja sivinske slike med oglišči $(100,50)$ in $(250,200)$. Prikažite originalno in povečano pravokotno področje slike. Kakšen je vpliv interpolacije na kakovost slike?

c. Podobno kot pri nalogi (b) izvedite (5x) povečavo pravokotnega področja barvne slike. Prikažite originalno in povečano pravokotno področje slike.

7. Napišite funkcijo za piramidno decimacijo sivinskih slik:

```
function oImage = decimateImage2D( iImage, iLevel ),
```

kjer je `iImage` vhodna sivinska slika, `iLevel` število zaporednih decimacij vhodne slike s faktorjem dva. Pred vsako decimacijo filtrirajte sliko z nizkoprepustnim sitom velikosti 3x3 (*glej opis piramidne decimacije*) s funkcijo `discreteConvolution2D()`. Funkcija vrne decimirano sivinsko sliko `oImage`.

a. Preizkusite delovanje funkcije na sivinski sliki za različne vrednosti `iLevel`, večje od 0. Izvedite decimacijo tudi brez filtriranja ter primerjajte dobljeni sliki. Kaj opazite?

b. Prilagodite funkcijo tako, da bo omogočala decimiranje barvnih slik in preizkusite delovanje na barvni sliki.

Vaja 4: Robustno iskanje 2D objektov

Gradient slike

Pri vaji bomo obravnavali postopke za iskanje oz. detekcijo oglišč, robov in črt v 2D slikah. S kombinacijo ali z ustreznimi nadgradnjami lahko te postopke uporabimo tudi za iskanje oz. detekcijo kompleksnejših objektov v 2D slikah. Velike lokalne spremembe sivinskih vrednosti na prehodih med različnimi objekti oz. področji 2D slik omogočajo iskanje oglišč, robov in črt na podlagi odvodov.

Prvi odvod določimo z velikostjo gradienta, ki ga za 2D funkcijo $f(x,y)$ zapišemo v obliki vektorja:

$$\nabla f(x,y) = \boldsymbol{g}(x,y) = \begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix}.$$

Vektorska slika gradienta $\boldsymbol{g}(x,y)$ v vsaki točki (x,y) vsebuje vektor, ki z x koordinatno osjo oklepa kot:

$$\alpha(x,y) = \arctan\frac{g_y(x,y)}{g_x(x,y)},$$

in kaže v smeri največje spremembe funkcije $f(x,y)$, torej pravokotno na rob. Velikost gradienta oz. dolžino vektorja izračunamo kot:

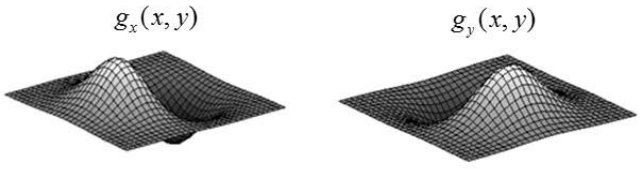
$$G(x,y) = \sqrt{g_x^2(x,y) + g_y^2(x,y)}.$$

Komponenti gradienta $g_x(x,y)$ in $g_y(x,y)$, ki predstavljata parcialna odvoda, na digitalni sliki izračunamo z digitalnimi filtri, naprimer s **Sobelovim operatorjem**. Odziv filtra dobimo s pomočjo 2D diskretne konvolucije med vhodno sivinsko sliko in operatorjem.

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

Zvezna oblika gradientnega operatorja



Zvezno obliko gradientnega operatorja, ki omogoča zasnovo poljubno velikega digitalnega filtra z nastavljivo stopnjo glajenja dobimo z odvajanjem Gaussove funkcije $N(x,y;\sigma)$:

$$\nabla N(x,y;\sigma) = \nabla e^{-\frac{x^2+y^2}{2\sigma^2}} = \begin{bmatrix} -x \\ -y \end{bmatrix} \frac{1}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Stopnjo glajenja poljubno izberemo s parametrom standardne deviacije σ.

Robovi v slikah imajo velik gradient v smeri pravokotno na rob, v ogliščih pa je gradient slike velik v več smereh. Oglišča so izraziti strukturni elementi slike oz. objektov na sliki in jih lahko uporabimo v aplikacijah kot so sledenje objektov v zaporednih posnetkih, določanje preslikave med različnimi pogledi enakega objekta, kot referečne točke za geometrijske meritve, za kalibracijo optičnih sistemov itd.

Harrisov detektor oglišč

Večina detektorjev oglišč temelji na primerjavi velikosti komponent gradienta v x in y smeri slike. V praksi se pogosto uporablja **Harrisov detektor oglišč**, pri katerem za vsako točko (x, y) izračunamo lokalno strukturno matriko $\boldsymbol{M}$:

$$\boldsymbol{M}(x,y) = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = \begin{bmatrix} g_x^2(x,y) & g_x(x,y)g_y(x,y) \\ g_x(x,y)g_y(x,y) & g_y^2(x,y) \end{bmatrix}.$$

Komponente lokalne strukturne matrike $\boldsymbol{M}$, tj. slike $A(x,y), B(x,y)$ in $C(x,y)$, nato zgladimo z Gaussovim jedrom $N(x,y;\sigma)$. Kriterijsko funkcijo za detekcijo oglišč v vsaki točki slike (x,y) izračunamo na osnovi lastnih vrednosti λ_1 in λ_2 matrike $\boldsymbol{M}$:

$$\lambda_{1,2} = \frac{\text{trace}(\boldsymbol{M})}{2} \pm \sqrt{\left(\frac{\text{trace}(\boldsymbol{M})}{2}\right)^2 - \det(\boldsymbol{M})},$$

kjer sta `trace()` in `det()` funkciji za izračun sledi in determinante matrike $\boldsymbol{M}$**.** Na področjih slike s konstantno sivinsko vrednostjo bo $\boldsymbol{M} = 0$ in $\lambda_1 = \lambda_2 = 0$, pri enakomerno naraščujoči sivinski vrednosti v eni smeri pa bo $\lambda_1 > 0$ in $\lambda_2 = 0$. Lastni vrednosti λ_1 in λ_2 torej kodirata izrazitost roba, pripadajoči lastnih vektorji pa smer roba. V ogliščih je rob izrazit v smerih obeh lastnih vektorjev, lastni vrednosti sta veliki in velja $|\lambda_1| > |\lambda_2|$, rob pa je tem bolj izrazit, čim manjša je razlika med lastnima vrednostima. Od tod sledi kriterijska funkcija za detekcijo oglišč:

$$Q(x,y) = \det(\boldsymbol{M}) - \alpha \left(\text{trace}(\boldsymbol{M})\right)^2.$$

Parameter α določa občutljivost detektorja in ga izberemo na intervalu $[0, \frac{1}{4}]$. Točke roba detektiramo kot lokalne maksimume v $Q(x,y)$, nadalje pa jih detektirana oglišča prečistimo z upragovljanjem $Q(x_i, y_i) > T_{min}$, kjer je T_{min} običajno na intervalu od $10^4 - 10^6$.

1. Napišite funkcijo za izračun odziva Harrisovega detektorja oglišč na vhodni sivinski sliki `iImage`:

   ```
   function oResponseQ = responseHarris(iImage, iAlpha,
                                        iSigma),
   ```

   kjer je `iAlpha` občutljivost detektorja oglišč, `iSigma` pa standardna deviacija Gaussove funkcije. Funkcija vrne sliko `oResponseQ`, ki predstavlja odziv detektorja oglišč.

   a. Naložite barvno RGB sliko *slika1.jpg* v okolju Matlab in jo pretvorite v sivinsko sliko po enačbi $S_i = 0{,}299R_i + 0{,}587G_i + 0{,}114B_i$. Preizkusite delovanje funkcije na sivinski sliki S_1 za različne vrednosti `iAlpha` in `iSigma` in obrazložite vpliv na sliko odziva `oResponseQ`.

2. Napišite funkcijo za iskanje lokalnih maksimumov v poljubnem 2D polju `iArray`:

   ```
   function oLocalMax = findLocalMax( iArray ).
   ```

   Funkcija vrne matriko `oLocalMax` z (x,y) koordinatami lokalnih maksimumov. Lokalni maksimumi v 2D polju so točke, v katerih je vrednost polja večja od vrednosti v sosednjih 8 elementih slike. Pri tem ne upoštevajte elementov slike, ki ležijo izven domene slike.

   a. Preizkusite delovanje funkcije na odzivu Harrisovega detektorja oglišč `oResponseQ`. Kateri od parametrov funkcije `responseHarris()`, `iAlpha` ali `iSigma` ima večji vpliv na število zaznanih lokalnih maksimumov?

3. Napišite funkcijo za detekcijo oglišč na vhodni sivinski sliki `iImage`:

   `function oCorners = cornerHarris( iImage, iAlpha, iThreshold ),`

   kjer je `iAlpha` občutljivost detektorja oglišč, `iThreshold` pa prag za detekcijo oglišč. Funkcija vrne matriko `oCorners` z (x, y) koordinatami detektiranih oglišč. Pri tem uporabite funkciji `responseHarris()` in `findLocalMax()`. Parameter `iSigma` v funkciji `responseHarris()` postavite na vrednost 3.

   a. Preizkusite delovanje funkcije na sivinski sliki S_1 in izberite po vašem optimalne vrednosti parametrov `iAlpha` in `iThreshold`. Naložite sliko *slika2.jpg* in jo pretvorite v sivinsko sliko S_2. Preverite delovanje funkcije še na sivinski sliki S_2. Sliki prikažite in s pomočjo ukazov `plot()` in `hold on` dorišite detektirana oglišča.

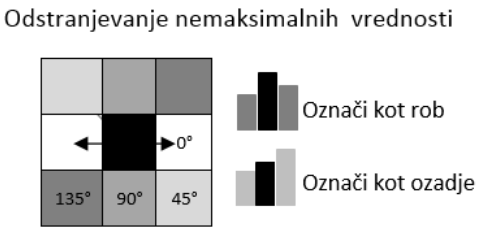
   b. Razširite funkcijo `cornerHarris()` tako, da bo imela dodatni vhodni parameter `iMinDist`. Parameter `iMinDist` določa minimalno razdaljo med oglišči. Prečistite seznam oglišč tako, da bodo oglišča med seboj oddaljena vsaj za razdaljo `iMinDist`, pri tem pa ohranite oglišča z večjim odzivom Harris detektorja. Preizkusite delovanje razširjene funkcije na sivinski sliki S_1.

Cannyev detektor robov

Zaznavanje robov v slikah je eden izmed najbolj pogostih postopkov na področju obdelave in analize slik, **Cannyev detektor robov** pa je eden izmed najpogosteje uporabljanih tovrstnih postopkov.

Cannyev detektor robov ima štiri osnove korake:

1. Glajenje slike z Gaussovim filtrom
2. Izračun slike velikosti in smeri gradienta
3. Odstranjevanje nemaksimalnih vrednosti
4. Dvojno upragovljanje in povezovanje robov



Za glajenje in določanje slik gradienta lahko uporabimo poljubni digitalni filter. V sliki gradienta dobimo okoli robov vedno širša področja v okolici največjih vrednosti odziva, ki jih moramo zožiti oz. odstraniti. **Odstranjevanje nemaksimalnih vrednosti** naredimo tako, da postavimo na nič vrednosti tistih slikovnih elementov, ki nimajo maksimalnih vrednosti v diskretni smeri gradienta (0°,45°,90° in 135°) oz. katerih sosednji slikovni elementi imajo v smeri gradienta večje vrednosti. V zadnjem koraku postopka izvedemo še upragovljanje slike robov in povezovanje robnih točk. Upragovljanje izvedemo z dvema pragoma. Z zgornjim pragom T_H določimo izrazite robne točke $r_H(x, y)$, ki jih bomo neposredno zadržali. S spodnjim pragom T_L določimo neizrazite robne točke $r_L(x, y)$, ki jih bomo obdržali za postopek povezovanja robov. Povezovanje izrazitih robov $r_H(x, y)$ izvedemo tako, da vsako sosednjo točko, ki pripada $r_L(x, y)$ vključimo v množico končnih robnih točk.

1. Napišite funkcijo za odstranjevanje nemaksimalnih vrednosti s pomočjo vhodnih slik gradientov `iGradX` in `iGradY`:

   `function oEdge = nonMaximaSuppression( iGradX, iGradY ).`

   Funkcija vrne binarno sliko `oEdge`, v kateri elementi z vrednostjo 1 predstavljajo nepovezane robove v sliki.

a. Preizkusite delovanje funkcije za sivinsko sliko S_1 tako, da gradientni sliki `iGradX` in `iGradY` izračunate z uporabo zveznega gradientnega operatorja. Obrazložite vpliv standardne deviacije σ na sliko robov.

2. Napišite funkcijo za upragovljanje in povezovanje robov v vhodni sliki `iEdge`:

```
function oEdge = connectEdge( iEdge, iGradX, iGradY, iThreshold ),
```

kjer sta `iGradX` in `iGradY` sliki gradienta, `iThreshold` pa dvovrstični vektor s pragoma T_L in T_H. Funkcija vrne sliko povezanih robov `oEdge`.

a. Preizkusite delovanje funkcije za sivinsko sliko S_1, in sicer na sliki robov z odstranjenimi nemaksimalnimi vrednostmi. Obrazložite vpliv vrednosti `iThreshold` na sliko robov.

b. Združite sliko robov s sivinsko sliko tako, da elemente, ki pripadajo robu, v sivinski sliki obarvate zeleno. Prikažite rezultirajočo sliko z uporabo sivinske slike S_1.

3. Napišite funkcijo za detekcijo robov na vhodni sivinski sliki `iImage`:

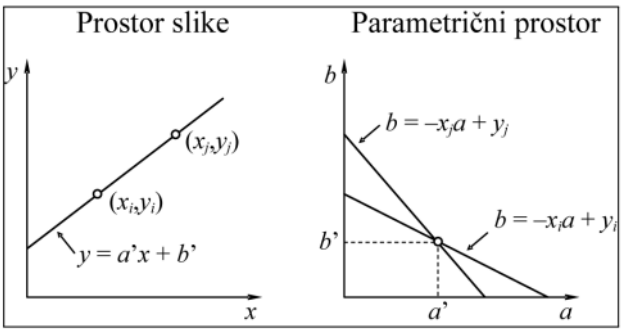
```
function oEdge = edgeCanny( iImage, iThreshold, iSigma ),
```

kjer je `iThreshold` dvovrstični vektor s pragoma T_L in T_H, `iSigma` pa standardna deviacija zveznega gradientnega operatorja. Funkcija vrne sliko povezanih robov `oEdge`. Pri tem uporabite funkciji `nonMaximaSuppression()` in `connectEdge()`.

a. Preverite pravilnost delovanja funkcije na sivinski sliki S_1 nato pa preizkusite še delovanje funkcije na sivinski sliki S_2. Ali je smiselno uporabiti enake vrednosti parametrov `iThreshold` in `iSigma` za detekcijo robov v prvi in drugi sliki?

Houghova preslikava

Med postopke iskanje objektov na sliki spada tudi iskanje premic oz. črt. Učinkovit postopek iskanja premic na sliki temelji na **Houghovi transformaciji oz. preslikavi**. Predpostavimo, da imamo binarno sliko robov $r(x, y)$. Skozi poljubno izbrano robno točko (x_i, y_i) lahko potegnemo poljubno število premic oblike:



$$y_i = ax_i + b.$$

Isto enačbo lahko zapišemo v prostoru parametrov (a, b), kjer sta koordinati robne točke x_i in y_i parametra premice:

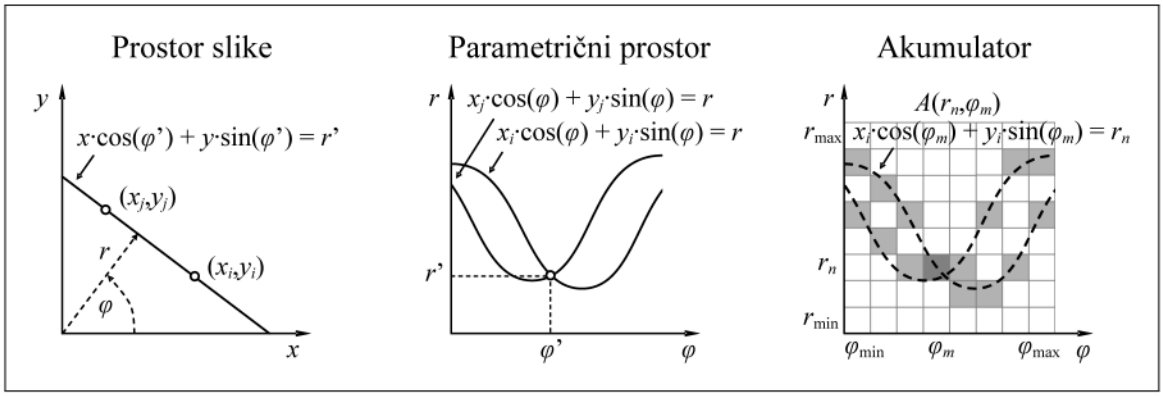
$$b = -x_i a + y_i.$$

Za neko drugo robno točko (x_j, y_j) dobimo v prostoru parametrov (a, b) še eno premico, ki se v neki točki (a', b') seka s premico, ki pripada (x_i, y_i). Parametra (a', b') določata enačbo premice v prostoru slike $r(x, y)$, ki gre skozi točki (x_i, y_i) in (x_j, y_j). Če za vse robne točke v sliki vnesemo pripadajočo premico v parametrični prostor, potem lahko poiščemo premice oz. črte z detekcijo vrhov v parametričnem prostoru.

Pri tovrstnemu zapisu enačbe premice naletimo na problem pri navpičnih premicah, pri katerih gre parameter $a \to \infty$, s tem pa tudi velikost parametričnega prostor(a, b). Temu se lahko izognemu z zapisom premice v polarnih koordinatah (r, φ):

$$x \cos\varphi + y \sin\varphi = r$$

Za navpično premico $(\varphi = 0°)$ določa r presečišče z x osjo, za vodoravno premico $(\varphi = 90°)$ pa določa r presečišče premice z y osjo. V polarnem prostoru prostoru (r, φ) vsaka sinusna krivulja $x_i \cos\varphi + y_i \sin\varphi = r$ predstavlja množico premic, ki gredo v prostoru slike (x, y) skozi točko (x_i, y_i). Presečišče dveh ali večih sinusnih krivulj (r', φ') določa parametra premice v prostoru (x, y).



Prostor parametrov (r, φ) lahko enostavno diskretiziramo, in sicer $0 \leq \varphi < \pi$ ter $-r_{diag} \leq r \leq r_{diag}$. Pri tem je $r_{diag} = \sqrt{M^2 + N^2}/2$ in M, N sta višina in širina slike $r(x, y)$, referenčna točka pa je center slike $(\frac{M}{2}, \frac{N}{2})$. Diskretiziran prostor imenujemo akumulator $A(r_n, \varphi_m)$, ki ga najprej postavimo na 0, nato pa za vsako robno točko (x_i, y_i) in za vsako možno diskretno vrednost $0 \leq \varphi_m < \pi$ izračunamo ustrezno diskretno vrednost parametra r_n, nato pa vrednost akumulatorja v celici (r_n, φ_m) povečamo za 1. Premice, ki predstavljajo robne točke slike, določimo tako, da poiščemo tiste celice akumulatorja $A(r_n, \varphi_m)$, ki imajo dovolj velike in lokalno največje vrednosti.

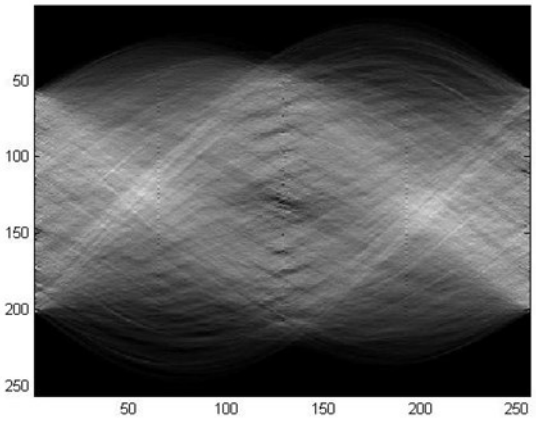
1. Napišite funkcijo, ki izračuna Houghov transform vhodne slike robov `iEdge`:

```
function oAcc = houghTransform( iEdge, iNumBinsR, iNumBinsPhi ),
```

   kjer `iNumBinsR` in `iNumBinsPhi` predstavljata število diskretnih predalov akumulatorja $A(r_n, \varphi_m)$. Funkcija vrne Houghov transform oz. akumulator `oAcc`.

   a. Preizkusite delovanje funkcije na sivinski sliki S_1 tako, da s Cannyjevim detektorjem roba izračunate vhodno sliko robov `iEdge`. Uporabite vrednosti `iNumBinsR = 256` in `iNumBinsPhi = 256` in preverite pravilnost delovanja s spodnjo sliko.

Houghov transform in sivinska slika S_1 z detektiranimi tremi najizrazitejšimi črtami



b. Daljši robovi bodo v akumulator doprinesli več sinusnih krivulj in posledično tudi več točk v akumulatorske celice. Zaradi tega bomo z večjo verjetnostjo detektirali daljše kot krajše robove v sliki. Temu se izognemo tako, da normaliziramo akumulator vhodne slike $A(r_n, \varphi_m)$. Dopolnite funkcijo `houghTransform()` tako, da izračunate akumulator $A_0(r_n, \varphi_m)$ za naključno generirano sliko robov $r_0(x, y)$, ki je enako velika kot vhodna slika robov `iEdge`. Dobljeni akumulator $A_0(r_n, \varphi_m)$ zgladite z Gaussovo funkcijo ter nato normalizirajte akumulator vhodne slike robov kot $A \leftarrow A/A_0$.

c. Od česa je odvisna natančnost določanja premic s pomočjo Houghove transformacije? Obrazložite odgovor.

2. Napišite funkcijo za iskanje poljubnega števila najizrazitejših premic oz. črt v sliki:

```
function oLines = findLines( iM, iN, iAcc, iNumLines )
```

kjer sta `iM` in `iN` višina in širina vhodne slike, `iAcc` Houghov transform slike robov in `iNumLines` število premic oz. črt v sliki, ki jih želimo detektirati. Uporabite funkcijo `findLocalMax()` za detekcijo lokalnih maksimumov v Houghovem transformu. Funkcija vrne matriko `oLines` z (r, φ) parametri detektiranih premic oz. črt dimenzije [`iNumLines` 2].

a. Preizkusite delovanje funkcije na Houghovih transformih ustreznih slik robov sivinskih slik S_1 in S_2. V sivinske slike vrišite detektirane premice v poljubni barvi s pomočjo funkcije `drawLine()` (Vaja 0). Preverite pravilnost delovanja za sliko S_1 z vrednostjo `iNumLines = 3` s pomočjo priložene slike, na kateri so detektirane premice vrisane z rdečo barvo.

b. Kako bi nadgradili Houghov transform tako, da bi omogočal detekcijo krajišč posameznih robov v vhodni sliki?

Vaja 5: Geometrijske preslikave in poravnava oblik

Geometrijska preslikava

Z geometrijskimi preslikavami $\boldsymbol{T}$: $R^2 \rightarrow R^2$ oz. $R^3 \rightarrow R^3$ preslikamo vse slikovne elemente 2D slik (x, y) oz. 3D slik (x, y, z) na nove lokacije (x', y') oz. (x', y', z'), njihove sivinske vrednosti pa pri tem ohranimo. Na ta način lahko izvedemo povečavo oz. pomanjšavo (skaliranje), premik (translacijo), zasuk (rotacijo), pa tudi številne druge linearne in nelinearne geometrijske preslikave slik. Poljubno geometrijsko preslikavo za 2D oz. 3D sliko zapišemo kot:

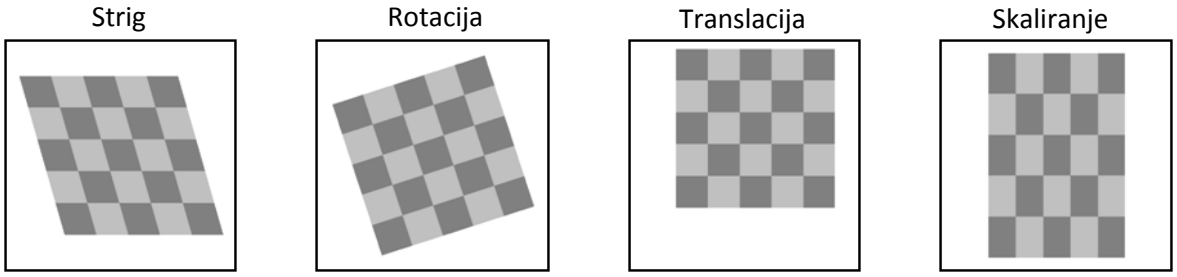
$$(x', y') = \boldsymbol{T}(x, y) \quad \text{oz.} \quad (x', y', z') = \boldsymbol{T}(x, y, z).$$

Najbolj splošna linearna geometrijska preslikava je **afina preslikava**, ki omogoča poljubno skaliranje, translacijo, rotacijo in strig. Afina preslikava je v 2D določena s 6, v 3D pa z 12 parametri:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \boldsymbol{T} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad \text{oz.} \quad \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \boldsymbol{T} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix},$$

kjer parametri t_x, t_y, t_z določajo translacijo v x, y, z smeri, parametri a_{ij} pa skaliranje, rotacijo in strig. Matriko za afino preslikavo lahko sestavimo z zaporednim množenjem homogenih matrik posameznih elementarnih preslikav v želenem vrstnem redu, naprimer:

$$\boldsymbol{T}_{afina} = \boldsymbol{T}_{strig}\, \boldsymbol{T}_{rot}\, \boldsymbol{T}_{trans}\, \boldsymbol{T}_{skal}.$$



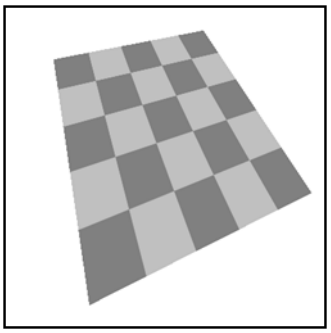
$$\boldsymbol{T}_{2D} = \begin{bmatrix} 1 & g_{xy} & 0 \\ g_{yx} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{T}_{2D} = \begin{bmatrix} c_\alpha & -s_\alpha & 0 \\ s_\alpha & c_\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{T}_{2D} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{T}_{2D} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{T}_{3D} = \begin{bmatrix} 1 & g_{xy} & g_{xz} & 0 \\ g_{yx} & 1 & g_{yz} & 0 \\ g_{zx} & g_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{T}_{3D} = \boldsymbol{T}_\gamma\, \boldsymbol{T}_\beta\, \boldsymbol{T}_\alpha \qquad \boldsymbol{T}_{3D} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{T}_{3D} = \begin{bmatrix} k_x & 0 & 0 & 0 \\ 0 & k_y & 0 & 0 \\ 0 & 0 & k_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

V praksi se pogosto uporabljata **toga preslikava**, ki jo dobimo z združevanjem rotacijske in translacijske elementarne preslikave in **podobnostna preslikava**, ki jo dobimo z združevanjem toge preslikave in skaliranja. Poleg linearnih se uporabljajo tudi nelinearne preslikave, kot npr. **projektivna preslikava**,

ki je v 2D določena z 8, v 3D pa s 15 parametri. Projektivno preslikavo zapišemo z nehomogeno matriko.

Projektivna preslikava



$$\boldsymbol{T}_{2D}:$$

$$x' = \frac{a_{11}x + a_{12}y + t_x}{p_x x + p_y y + 1}$$

$$y' = \frac{a_{21}x + a_{22}y + t_y}{p_x x + p_y y + 1}$$

$$\boldsymbol{T}_{3D}:$$

$$x' = \frac{a_{11}x + a_{12}y + a_{13}z + t_x}{p_x x + p_y y + p_z z + 1}$$

$$y' = \frac{a_{21}x + a_{22}y + a_{23}z + t_y}{p_x x + p_y y + p_z z + 1}$$

$$z' = \frac{a_{31}x + a_{32}y + a_{33}z + t_z}{p_x x + p_y y + p_z z + 1}$$

Za izvedbo vaje boste potrebovali zbirko 2D oblik, ki je javno dostopna in si jo naložite s spletne strani http://visionlab.uta.edu/shape_data/hmm_gpd.zip, 2D sliko triglav.jpg in zbirko 3D oblik glave.mat. Za prikazovanje 3D oblik boste potrebovali še ukaz renderSurface.m.

1. Napišite funkcijo, ki ustvari matriko poljubne 2D afine preslikave:

   ```
   function oMat2D = transAffine2D( iScale, iTrans, iRot,
                                    iShear ),
   ```

   kjer je `iScale` dvovrstični vektor s parametroma skale k_x, k_y, `iTrans` dvovrstični vektor s parametroma translacije t_x, t_y, `iRot` kot rotacije α v stopinjah in `iShear` dvovrstični vektor s parametroma striga g_x, g_y. Funkcija vrne homogeno matriko `oMat2D`, ki ima dimenzije 3×3.

   a. Iz zbirke 2D oblik naložite 2D obliko *plane_data/Class1_Sample1.mat*. S pomočjo funkcije ustvarite poljubne 2D afine preslikave in preslikajte dano 2D obliko. Originalno in preslikane oblike narišite z ukazoma `plot()` in `hold on`.

   b. Z zaporednim množenjem ustreznih matrik zagotovite, da se bo 2D oblika rotirala okoli svojega centra oz. aritmetične sredine.

2. Napišite funkcijo, ki ustvari matriko poljubne 2D projektivne preslikave:

   ```
   function oMat2D = transProjective2D( iAffineMat2D, iProj ),
   ```

   kjer je `iAffineMat2D` matrika poljubne 2D afine preslikave, `iProj` pa dvovrstični vektor s parametroma projekcije p_x, p_y. Funkcija vrne nehomogeno matriko `oMat2D` dimenzij 3×3.

   a. S pomočjo funkcije ustvarite poljubne 2D projektivne preslikave in preslikajte dano 2D obliko. Pri tem zagotovite, da boste koordinate (x, y) po projektivni preslikavi ustrezno pretvorili v homogene koordinate (x', y').

3. Napišite funkcijo za poljubno 2D afino (in projektivno) preslikavo vhodne sivinske slike `iImage`:

   ```
   function oImage = transImage2D( iImage, iMat2D ),
   ```

   kjer je `iMat2D` poljubna 2D preslikava. Funkcija vrne preslikano sivinsko sliko `oImage`. Pri tem uporabite funkcijo za interpolacijo prvega reda `interpolate1Image2D()` (Vaja 3).

a. Naložite barvno RGB sliko *triglav.jpg* v okolju Matlab in jo pretvorite v sivinsko sliko po enačbi $S = 0{,}299R + 0{,}587G + 0{,}114B$. Preizkusite delovanje funkcije za različne 2D afine in projektivne preslikave.

4. Napišite funkcijo, ki ustvari matriko poljubne 3D afine preslikave:

```
function oMat3D = transAffine3D( iScale, iTrans, iRot, iShear ),
```

kjer je `iScale` trivrstični vektor s parametri skale k_x, k_y, k_z, `iTrans` trivrstični vektor s parametri translacije t_x, t_y, t_z, `iRot` trivrstični vektor s koti rotacije α, β, γ v stopinjah in `iShear` šestvrstični vektor s parametri striga $g_{xy}, g_{xz}, g_{yx}, g_{yz}, g_{zx}, g_{zy}$. Funkcija vrne homogeno matriko `oMat3D`, ki ima dimenzije 4×4.

a. Iz zbirke 3D oblik *glave.mat* naložite 3D obliko `x=glave{1}`. S pomočjo funkcije ustvarite poljubne 3D afine preslikave in preslikajte dano 3D obliko. Originalno in preslikane oblike narišite z ukazom `renderSurface()`.

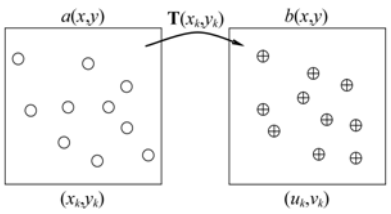
Določanje parametrov preslikave

Geometrijska poravnava dveh oblik je proces iskanja optimalne geometrijske preslikave $\boldsymbol{T}$, ki obliki preslika tako, da se iste strukture oz. značilnice oblike nahajajo v enakem ali čim bolj »podobnem« položaju. Značilnice 2D oz. 3D oblik so pogosto kar oblaki točk, zato se za mero podobnosti med dvema oblikama pogosto uporablja kar povprečna kvadratna Evklidska razdalja med pripadajočimi oz. korespondenčnimi točkami. Na osnovi množice K parov korespondenčnih točk lahko določimo preslikavo $\boldsymbol{T}$ tako, da se preslikane točke referenčne oblike $\boldsymbol{T}(x_i, y_i)$ čim boljše prilegajo točkam vhodne oblike (x_i', y_i') oz. obratno:

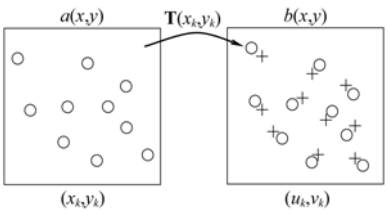
$$(x_i', y_i') \leftrightarrow \boldsymbol{T}(x_i, y_i) \quad \text{oz.} \quad (x_i, y_i) \leftrightarrow \boldsymbol{T}^{-1}(x_i', y_i').$$

Glede na število parov korespondenčnih točk in vrsto preslikave lahko točke lahko preslikamo tako, da se popolnoma prekrivajo, tj. $\boldsymbol{T}(x_i, y_i) = (x_i', y_i')$, kar imenujemo **interpolacijska poravnava**, ali pa tako, da se točke le približno prekrijejo, tj. $\boldsymbol{T}(x_i, y_i) \approx (x_i', y_i')$, kar imenujemo **aproksimacijska poravnava**.

Interpolacijska poravnava



Aproksimacijska poravnava



Afino interpolacijsko poravnavo med 2D oblikami lahko enolično določimo, če poznamo tri pare ($K = 3$) nekolinearnih točk $(x_i', y_i') \leftrightarrow (x_i, y_i)$, med 3D oblikami pa če poznamo šest parov ($K = 6$) nekolinearnih točk. V 2D lahko preslikavo $\boldsymbol{T}$ med tremi pari točk zapišemo kot:

$$\boldsymbol{T} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_1' & x_2' & x_3' \\ y_1' & y_2' & y_3' \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1}.$$

Afino aproksimacijsko poravnavo lahko med 2D oblikami izvedemo, kadar poznamo več kot tri pare ($K > 3$), med 3D oblikami pa več kot šest parov ($K > 6$) korespondenčnih točk. V tem primeru dobimo predoločen sistem enačb, zato korespondenčne točke lahko poravnavo le približno (aproksimacijsko).

To storimo tako, da minimiziramo povprečno kvadratno Evklidsko razdaljo med korespondenčnimi točkami:

$$\Sigma^2 = \frac{1}{K}\sum_{i=1}^{K}\|(x_i', y_i') - \boldsymbol{T}(x_i, y_i)\|^2.$$

Optimalne vrednosti neznanih parametrov preslikave dobimo tako, da odvode povprečne kvadratne Evklidske razdalje Σ^2 po vseh parametrih postavimo na nič in dobimo sistem enačb za reševanje. Za 2D oblike dobimo naslednji sistem enačb:

$$\begin{bmatrix} \overline{xx} & \overline{xy} & \bar{x} & 0 & 0 & 0 \\ \overline{xy} & \overline{yy} & \bar{y} & 0 & 0 & 0 \\ \bar{x} & \bar{y} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \overline{xx} & \overline{xy} & \bar{x} \\ 0 & 0 & 0 & \overline{xy} & \overline{yy} & \bar{y} \\ 0 & 0 & 0 & \bar{x} & \bar{y} & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ t_x \\ a_{21} \\ a_{22} \\ t_y \end{bmatrix} = \begin{bmatrix} \overline{x'x} \\ \overline{x'y} \\ \overline{x'} \\ \overline{y'x} \\ \overline{y'y} \\ \overline{y'} \end{bmatrix},$$

kjer elementi s črto označujejo povprečne vrednosti $\overline{xy'} = \frac{1}{K}\sum_{k=1}^{K} x_i\, y_i'$ oz. $\bar{x} = \frac{1}{K}\sum_{k=1}^{K} x_i$. Zgornji sistem lahko zapišemo v matrični obliki in vektor t neznanih parameterov afine preslikave $\boldsymbol{T}$ zapišemo kot $\boldsymbol{P}_{xy}\,\boldsymbol{t} = \boldsymbol{p}_{x'y'} \;\rightarrow\; \boldsymbol{t} = \boldsymbol{P}_{xy}^{-1}\boldsymbol{p}_{x'y'}$.

Toga poravnava, ki se pogosto uporablja v praski, je vedno aproksimacijska, saj zahtevano minimalno število parov korespondenčnih točk vodi do predoločenega sistema enačb – dobimo namreč več enačb kot pa je neznanih parametrov. Za poravnavo 2D oblik potrebujemo $K \geq 2$ korespondenčnih točk, preslikavo pa določimo z minimizacijo povprečne kvadratno Evklidske razdaljo med točkami:

$$\Sigma^2 = \frac{1}{K}\sum_{i=1}^{K}\Big((x_i\cos\alpha - y_i\sin\alpha + t_x - x_i')^2 + \big(x_i\sin\alpha + y_i\cos\alpha + t_y - y_i'\big)^2\Big),$$

ki jo odvajamo po parametrih t_x, t_y, α in odvode postavimo na nič. Rešitev sistema enačb je:

$$\alpha = -\text{arctg}\,\frac{\overline{x'y} - \overline{y'x} - \overline{x'}\,\bar{y} + \overline{y'}\,\bar{x}}{\overline{x'x} + \overline{y'y} - \overline{x'}\,\bar{x} - \overline{y'}\,\bar{y}},$$
$$t_x = \overline{x'} - \bar{x}\cos\alpha + \bar{y}\sin\alpha\ ,$$
$$t_y = \overline{y'} - \bar{x}\sin\alpha - \bar{y}\cos\alpha\ .$$

Za poravnavo 3D oblik potrebujemo $K \geq 3$ korespondenčnih točk, obstaja pa več numeričnih postopkov za določanje preslikave $\boldsymbol{T}$. Arun in dr. so predlagali postopek, ki minimizira Σ^2 temelji na ujemanju povprečnega položaja točk po poravnavi dveh 3D oblik, zato lahko Σ^2 izrazimo le z rotacijsko matriko $\boldsymbol{R}$:

$$\Sigma^2 = \frac{1}{K}\sum_{i=1}^{K}\|\boldsymbol{d_i^c} - \boldsymbol{R}\boldsymbol{p_i^c}\|^2,$$

kjer so $\boldsymbol{d_i^c} = (x_i', y_i', z_i') - \overline{\boldsymbol{d}}$ in $\boldsymbol{p_i^c} = (x_i, y_i, z_i) - \overline{\boldsymbol{p}}$ centrirane korespondenčne točke referenčne in vhodne 3D oblike. Rotacijsko matriko $\boldsymbol{R}$, ki minimizira gornji izraz dobimo s pomočjo korelacijske matrike $\boldsymbol{H}$ in razcepom na singularne vrednosti (v Matlabu ukaz `svd( )`):

$$\boldsymbol{H} = \sum_{i=1}^{K}\boldsymbol{p_i^c}\,\boldsymbol{d_i^{c\mathrm{T}}} \;\overset{SVD}{\longrightarrow}\; \boldsymbol{H} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{V}^{\mathrm{T}} \;\rightarrow\; R = \boldsymbol{V}\boldsymbol{U}^{\mathrm{T}}.$$

Determinanta rotacijske matrike mora biti enaka +1, sicer rešitev popravimo v $\boldsymbol{R} = \boldsymbol{V}'\boldsymbol{U}^{\mathrm{T}}$, kjer je $\boldsymbol{V}' = [v_1, v_2, -v_3]$ in v_3 je stolpec matrike $\boldsymbol{V}$ singularni vrednosti 0 matrike $\boldsymbol{H}$. Paramere translacije $\boldsymbol{t}$ določimo iz razlike med povprečnima položajema točk po rotaciji, tj. $\boldsymbol{t} = \bar{\boldsymbol{d}} - \boldsymbol{R}\bar{\boldsymbol{p}}$.

5. Napišite funkcijo, ki med pari korespondenčnih točk določi 2D afino interpolacijsko poravnavo:

   `function oMat2D = mapAffineInterp2D( iPtsRef, iPtsMov ),`

   kjer je `iPtsRef` matrika s koordinatami referenčnih točk, `iPtsMov` pa matrika s koordinatami premičnih točk. Obe matriki sta dimenzij 3×2. Funkcija vrne 2D afino preslikavo v obliki homogene matrike `oMat2D`, ki ima dimenzije 3×3.

   a. Iz zbirke 2D oblik naložite 2D obliko *plane_data/Class1_Sample1.mat*. Naključno izberite tri točke v 2D obliki in jih preslikajte v korespondenčne pare točk s poljubno 2D afino preslikavo, ki jo ustvarite s funkcijo `transAffine2D()`. S korespondenčnimi pari točk določite 2D afino interpolacijsko poravnavo. Primerjajte elemente matrike izbrane poljubne 2D afine preslikave z elementi dobljene matrike. Obrazložite ugotovitve.

6. Napišite funkcijo, ki med pari korespondenčnih točk določi 2D afino aproksimacijsko poravnavo:

   `function oMat2D = mapAffineApprox2D( iPtsRef, iPtsMov ),`

   kjer je `iPtsRef` matrika s koordinatami referenčnih točk, `iPtsMov` pa matrika s koordinatami premičnih točk. Obe matriki sta dimenzij $K \times 2$, pri čemer je $K > 3$. Funkcija vrne 2D afino preslikavo v obliki homogene matrike `oMat2D`, ki ima dimenzije 3×3.

   a. Iz zbirke 2D oblik naložite dve 2D obliki, referenčno *plane_data/Class1_Sample1.mat* in premično *plane_data/Class1_Sample8.mat*. Izberite naključno monotono naraščujoče zaporedje 100 indeksov točk v referenčni obliki. Z izbranimi indeksi izluščite korespondenčne pare točk v referenčni in premični obliki in z njimi določite 2D afino aproksimacijsko poravnavo. Preslikajte vse točke premične oblike z dobljeno 2D afino preslikavo. Izrišite preslikano premično obliko in referenčno obliko ter ju primerjajte. Od česa zavisi kvaliteta poravnave med referenčno in preslikano premično obliko?

   b. Izberite naključno monotono naraščujoče zaporedje 3 indeksov točk v prvi obliki. Z izbranimi indeksi izluščite korespondenčne pare točk v referenčni in premični obliki in z njimi določite 2D afino interpolacijsko poravnavo. Preslikajte vse točke premične oblike z dobljeno 2D afino preslikavo. Izrišite preslikano premično obliko in referenčno obliko ter ju primerjajte. Obrazložite razliko v kvaliteti poravnave med referenčno in preslikano premično obliko za 2D afino interpolacijsko in aproksimacijsko poravnavo.

7. Napišite funkcijo, ki med pari korespondenčnih točk določi 2D togo aproksimacijsko poravnavo:

   `function oMat2D = mapRigid2D( iPtsRef, iPtsMov ),`

   kjer sta `iPtsRef` in `iPtsMov` matriki s koordinatami referenčnih in premičnih točk. Obe matriki imata dimenzije $K \times 2$, pri čemer je $K \geq 2$. Funkcija vrne 2D togo preslikavo v obliki homogene matrike `oMat2D`, ki ima dimenzije 3×3.

a. Korespondenčne pare točk v referenčni in premični obliki, ki ste jih določili pri nalogi 6.a uporabite za določitev 2D toge aproksimacijske poravnave. Preslikajte vse točke premične oblike z dobljeno 2D afino preslikavo ter izrišite preslikano premično obliko in referenčno obliko. Obrazložite razliko v kvaliteti poravnave med referenčno in preslikano premično obliko za 2D afino aproksimacijsko in 2D togo aproksimacijsko poravnavo.

8. Napišite funkcijo, ki med pari korespondenčnih točk določi 3D togo aproksimacijsko poravnavo:

```
function oMat3D = mapRigid3D( iPtsRef, iPtsMov ),
```

kjer sta `iPtsRef` in `iPtsMov` matriki s koordinatami refenčnih in premičnih točk. Obe matriki imata dimenzije $K \times 3$, pri čemer je $K \geq 3$. Funkcija vrne 3D togo preslikavo v obliki homogene matrike `oMat3D`, ki ima dimenzije 4×4.

a. Iz zbirke 3D oblik *glave.mat* naložite 3D obliko `x=glave{1}`. Naključno izberite 100 točk v 3D obliki in jih preslikajte v korespondenčne pare točk s poljubno 3D togo preslikavo, ki jo ustvarite s funkcijo `transAffine3D()`. S korenspondenčnimi pari točk določite 3D togo interpolacijsko poravnavo. Primerjajte elemente matrike izbrane poljubne 3D toge preslikave z elementi dobljene matrike. Obrazložite ugotovitve.

Pri poravnavi oblik korespondenčne točke običajno niso vnaprej znane in jih je potrebno določiti v procesu poravnave oblik. Ena od uveljavljenih metod za določanje korespondenčnih točk je metoda **iterativno najbližja točka** (*ang. iterative closest point - ICP*), pri kateri v vsakem koraku k določimo korespondenčne točke kot najbližje pare točk $(x_i', y_i') \leftrightarrow (x_i, y_i)$ in nato izračunamo interpolacijsko ali aproksimacijsko preslikavo $\boldsymbol{T}_k$ med temi pari točk. Premično obliko preslikamo s $\boldsymbol{T}_k$ in ponovimo korak. Postopek ponavljamo dokler je relativna sprememba $\|\boldsymbol{T}_k - \boldsymbol{I}\|_\infty > \varepsilon$ oz. dokler ne izvedemo maksimalno število korakov k_{max}. Če so preslikave $\boldsymbol{T}_k$ linearne, potem lahko preslikavo med oblikama po k-tem koraku določimo z zaporednim matričnim množenjem vmesnih preslikav:

$$\boldsymbol{T} = \boldsymbol{T}_k \boldsymbol{T}_{k-1} \cdots \boldsymbol{T}_1.$$

9. Napišite funkcijo za togo aproksimacijsko poravnavo dveh 2D oz. 3D oblik s postopkom ICP:

```
function [oMat, oErr] = alignRigidICP(
   iPtsRef, iPtsMov, iEps, iMaxIter ),
```

kjer sta `iPtsRef` in `iPtsMov` matriki s koordinatami referenčnih in premičnih točk. Obe matriki imata dimenzije $K \times D$. `iEps` in `iMaxIter` določata pogoje za zaustavitev postopka, kjer je `iEps` najmanjša sprememba parametrov preslikava, `iMaxIter` pa maksimalno število korakov k_{max}. Funkcija vrne 2D oz. 3D togo preslikavo v obliki homogene matrike `oMat` in vektor `oErr` velikosti $k_{max} \times 1$, ki podaja napako Σ^2 med korespondenčnimi točkami v vsakem koraku.

a. Iz zbirke 2D oblik naložite 2D oblike *plane_data/Class1_Sample**X**.mat*, ***X***=1,...30. Izvedite poravnavo vseh 2D oblik na obliko ***X***=1. Izrišite v eno sliko vse 2D oblike v začetni legi, v drugo sliko pa v poravnani legi. Od česa zavisi uspešnost postopka poravnave oblik?

b. Iz zbirke 3D oblik *glave.mat* naložite 3D oblike `x=glave{`***X***`}`, ***X***=1,...5. Izvedite poravnavo vseh 3D oblik na obliko ***X***=1. Izrišite in obrazložite potek napake Σ^2 za primer uspešne in neuspešne poravnave oblik.

Vaja 6: Poravnava slik za iskanje predlog

Križno-korelacijski koeficient

Iskanje predlog v slikah (ang. *template matching*) je posebna oblika poravnave slik, pri kateri želimo v vhodni sliki $r(x,y)$ najti predlogo oz. sliko objekta zanimanja $t(x,y)$. Iskanje predlog se pogosto uporablja pri kontroli kakovosti izdelkov, za navigacijo mobilnih robotov oz. za zaznavo poljubnih, vnaprej znanih objektov v vhodnih slikah. Cilj teh postopkov je določiti natančen položaj predloge v vhodni sliki, običajno s translacijo predloge v x in y smeri slike. Postopki za iskanje predlog običajno temeljijo na enostavnih merah podobnosti, ki jih lahko hitro in učinkovito ovrednotimo brez uporabe numeričnih optimizacijskih postopkov. Pri vaji boste za iskanje predloge za mero podobnosti uporabili **križno korelacijo slik** in njene izpeljanke. Križna korelacija slike $r(x,y)$ in predloge $t(x,y)$ je definirana kot:

$$\varphi(u,v) = \sum_{i=1}^{I}\sum_{j=1}^{J} r(x_i+u, y_j+v)\cdot t(x_i, y_j)\ ,$$

kjer sta I in J dimenziji predloge t v x in y smeri, parametra u,v pa predstavljata translacijo predloge v x in y smeri. Vhodna slika $r(x,y)$ ima dimenzije N in M, pri čemer naj velja $N>I$ in $M>J$. Predlogo $t(x,y)$ v vhodni sliki $r(x,y)$ poiščemo tako, da ovrednotimo križno korelacijo na nekem območju parametrov u,v in nato izberemo optimalne parametre u^*,v^*, pri katerih je križna korelacija $\varphi(u,v)$ maksimalna.

Križna korelacija $\varphi(u,v)$ deluje zanesljivo ob predpostavki, da so sivinske vrednosti vhodne slike $r(x,y)$ in predloge $t(x,y)$ povsem linearno odvisne, kar pri realnih slikah pogosto ne velja. Ena od popularnih mer podobnosti, ki temelji na križni korelaciji in je robustna na lokalne linearne preslikave sivinskih vrednosti, tj. $r'(x,y) = a(x,y)\cdot r(x,y) + b(x,y)$, je **korelacijski koeficient**:

$$CC(u,v) = \frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(r(x_i+u,y_j+v)-\bar{r}(u,v)\right)\left(t(x_i,y_j)-\bar{t}\right)}{\sqrt{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(r(x_i+u,y_j+v)-\bar{r}(u,v)\right)^2\cdot\sum_{i=1}^{I}\sum_{j=1}^{J}\left(t(x_i,y_j)-\bar{t}\right)^2}}\ .$$

V imenovalcu opazimo izraza za standardno deviacijo vhodne slike $r(x,y)$ in predloge $t(x,y)$, neposredno povezavo med $\varphi(u,v)$ in $CC(u,v)$ pa dobimo tako, da razčlenimo izraz v števcu:

$$\sum_{i=1}^{I}\sum_{j=1}^{J} r(x_i+u,y_j+v)\cdot t(x_i,y_j) - \bar{r}(u,v)\cdot\bar{t} = \varphi(u,v) - \bar{r}(u,v)\cdot\bar{t}\ .$$

Vrednosti korelacijskega koeficienta $CC(u,v)$ ležijo na intervalu [-1, 1], pri čemer vrednost 1 predstavlja maksimalno podobnost med vhodno sliko $r(x,y)$ in predlogo $t(x,y)$.

Za izvedbo vaje boste uporabili slike *team**X**.jpg, **X**=1,2,3* in datoteko *predloge.mat*, ki vsebuje slike 11 predlog v spremenljivki *T*. Predloge *T* so bile izluščene iz slike *team1.jpg*, ustrezne x in y koordinate teh predlog najdete v spremenljivkah *xf* in *yf*.

1. Napišite funkcijo, ki izračuna *polje križne korelacije* med vhodno sliko `iImage` in predlogo `iTemplate`, in sicer za vse možne položaje predloge na področju vhodne slike:

   ```
   function oSim = matchCrossCorr( iImage, iTemplate ).
   ```

2. `iImage` in `iTemplate` sta sivinski sliki velikosti $N\times M$ in $I\times J$. Funkcija vrne spremenljivko `oSim` v obliki polja križne korelacije velikosti $N\times M$. Polje križne korelacije dobite tako, da

ovrednotite križno korelacijo $\varphi(u,v)$ za vse pare parametrov $(u,v)=\{u=1,...,N;v=1,...,M\}$. Predpostavite, da so sivinske vrednosti izven območja vhodne slike enaka 0 v primeru ko je del predloge izven vhodne slike.

   a. Naložite sliko *team1.jpg* in predloge *T* in jih pretvorite v ustrezne sivinske slike. Preizkusite delovanje funkcije za različne predloge tako, da izrišete polje križne korelacije in označite položaj maksimuma križne korelacije. Ali najdete položaj maksimuma na ustreznem mestu? Podajte nekaj primerov in obrazložite rezultate.

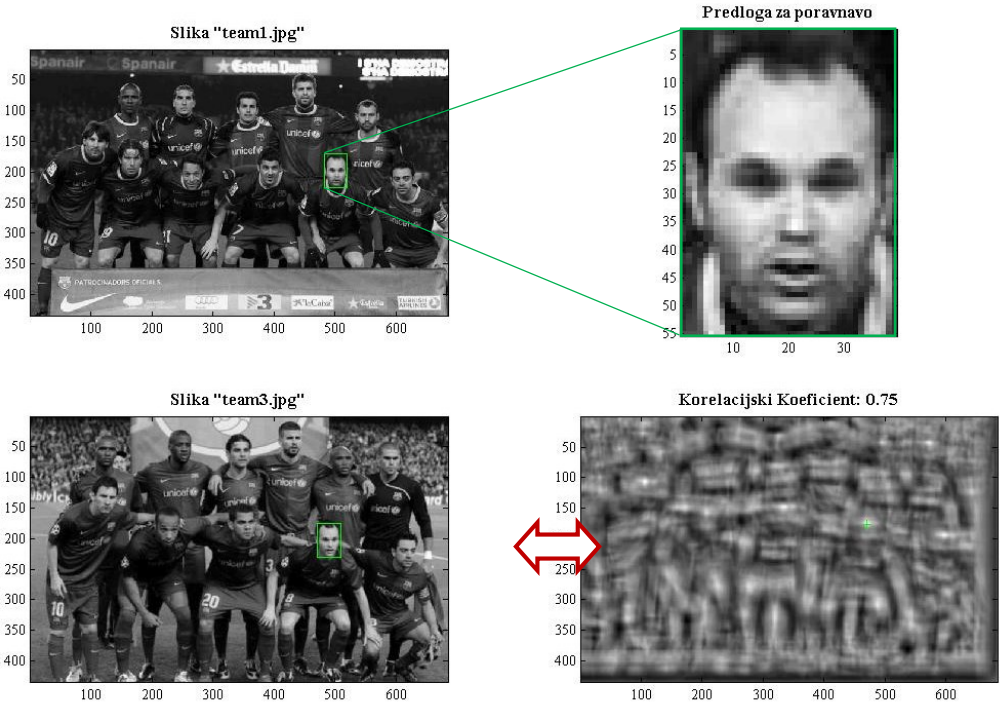
3. Napišite funkcijo, ki izračuna *polje korelacijskega koeficienta* med vhodno sliko `iImage` in predlogo `iTemplate`, in sicer za vse možne položaje predloge na področju vhodne slike:

   `function oSim = matchCorrCoeff( iImage, iTemplate ).`

4. `iImage` in `iTemplate` sta sivinski sliki velikosti $N\times M$ in $I\times J$. Funkcija vrne spremenljivko `oSim` v obliki polja korelacijskega koeficienta velikosti $N\times M$, podobno kot pri prejšnji nalogi.

   a. Preizkusite delovanje funkcije podobno kot v nalogi (1.a). Ali najdete položaj maksimuma na ustreznem mestu? Podajte nekaj primerov in obrazložite rezultate.

   b. Naložite sliki *team2.jpg* in *team3.jpg* in preizkusite delovanje poravnave za različne predloge. Za lažjo verifikacijo delovanja lahko v optimalnem položaju predloge u^*,v^*, ki ga dobite pri maksimalni vrednosti $CC(u,v)$, v vhodno sliko narišete ustrezen okvir v velikosti predloge, kot je prikazano na spodnji sliki:



Izračun v frekvenčnem prostoru

Postopke iskanja predloge, ki temeljijo na križni korelaciji slik, lahko zelo učinkovito izračunamo v frekvenčnem prostoru z uporabo **Fourierjeve transformacije**, ki je definirana kot:

$$F\{r(x,y)\}=R(\lambda_x,\lambda_y)=\frac{1}{\sqrt{NM}}\sum_{x=1}^{N}\sum_{y=1}^{M}r(x,y)\cdot e^{-i2\pi(\frac{x\lambda_x}{N}+\frac{y\lambda_y}{M})},$$

kjer je $F\{r(x,y)\} = R(\lambda_x, \lambda_y)$ Fourierjeva transformiranka vhodne slike $r(x,y)$, λ_x, λ_y sta koordinatni osi v frekvenčnem prostoru dimenzije $N \times M$. Inverzna Fourierjeva transformacija iz frekvenčnega v slikovni prostor je definirana kot:

$$F^{-1}\{R(\lambda_x, \lambda_y)\} = r(x,y) = \frac{1}{\sqrt{NM}} \sum_{\lambda_x=1}^{N} \sum_{\lambda_y=1}^{M} R(\lambda_x, \lambda_y) \cdot e^{+i2\pi(\frac{x\lambda_x}{N} + \frac{y\lambda_y}{M})}.$$

Fourierjeva in inverzna Fourierjeva transformacija se razlikujeta le po domeni izračuna in po predznaku v eksponentni funkciji. S Fourierjevo transformacijo 2D sliko predstavimo kot linearno kombinacijo diskretnih 2D sinusnih funkcij s horizontalnimi in vertikalnimi frekvencami λ_x, λ_y:

$$e^{-i2\pi(\frac{x\lambda_x}{N} + \frac{y\lambda_y}{M})} = \cos\left[2\pi(\frac{x\lambda_x}{N} + \frac{y\lambda_y}{M})\right] + i \cdot \sin\left[2\pi(\frac{x\lambda_x}{N} + \frac{y\lambda_y}{M})\right].$$

Vse diskretne 2D sinusne funkcije so periodične z največ $N/2$ oz. $M/2$ ponovitvami v horizontalni in vertikalni smeri slike. Za realne signale kot so slike bo $F\{r(x,y)\} = R(\lambda_x, \lambda_y)$ kompleksna spremenljivka. Amplitudo in fazo posameznih komponent diskretnih 2D sinusnih funkcij določimo kot:

$$A\left(R(\lambda_x, \lambda_y)\right) = \sqrt{R(\lambda_x, \lambda_y) \cdot \overline{R(\lambda_x, \lambda_y)}}, \quad \phi\left(R(\lambda_x, \lambda_y)\right) = \operatorname{arctg} \frac{\operatorname{Im}\left[R(\lambda_x, \lambda_y)\right]}{\operatorname{Re}\left[R(\lambda_x, \lambda_y)\right]}.$$

1. Napišite funkcijo za izračun Fourierjeve transformacije vhodne sivinske slike `iImage`:

   ```
   function oFT = computeFourierTransform( iImage ),
   ```

2. `iImage` ima velikost $N \times M$. Funkcija vrne Fourierjevo transformacijo vhodne slike v spremenljivki `oFT`, velikosti $N \times M$.

   a. Naložite sivinsko sliko predloge `T{1}` in preizkusite delovanje funkcije. Rezultat vaše funkcije primerjajte z rezultatom, ki ga dobite s klicem Matlabove funkcije `fft2()`.

   b. Izračunajte amplitude in faze diskretnih sinusnih funkcij in jih prikažite kot slike. Amplitudo preslikajte v logaritemsko merilo (`log()`), za prikaz amplitude in faze pa uporabite ukaz `imagesc(fftshift(.))`. Na primeru slik amplitude in faze obrazložite lastnosti Fourierjeve transformacije realnih signalov. Realne in imaginarne komponente transformacije dobite z ukazoma `real()` in `imag()`.

3. Dopolnite funkcijo `computeFourierTransform()` tako, da bo imela dodaten vhodni parameter `iDirection`, ki bo določal smer Fourirerjeve transformacije (direktna oz. inverzna). Npr., `iDirection` je lahko skalar z vrednostjo 1 oz. -1.

   a. Izračunajte inverzno Fourierjevo transformacijo na $R(\lambda_x, \lambda_y) = F\{r(x,y)\}$ iz (3.a). Rezultat primerjajte z rezultatom, ki ga dobite s klicem Matlabove funkcije `ifft2()`.

   b. Prikažite in primerjajte originalno sivinsko sliko predloge `T{1}` in sliko, ki ste jo dobili po inverzni Fourierjevi transformaciji v nalogi (4.a). Ali se sliki razlikujeta?

Fourierjevo transformacijo lahko izkoristimo za učinkovit izračun križne korelacije slik, pri tem pa se opiramo na dve pomembni lastnosti Fourierjeve transformacije:

1) **teorem o premiku** (ang. *shift theorem*):

$$F\left\{r(x+t_x, y+t_y)\right\} = F\left\{r(x,y)\right\} \cdot e^{-2\pi i(t_x\lambda_x + t_y\lambda_y)}$$

2) **teorem o konvoluciji** (ang. *convolution theorem*):

$$F\left\{r(x,y) * t(x,y)\right\} = F\left\{r(x,y)\right\} \cdot F\left\{t(x,y)\right\}$$

S pomočjo teh lastnosti lahko križno korelacijo $\varphi(u,v)$ v frekvenčnem prostoru zapišemo kot:

$$F\left\{\varphi(u,v)\right\} = \Gamma(\lambda_u, \lambda_v) = F\left\{r(x_i, y_j)\right\} \cdot \overline{F\left\{t(x_i, y_j)\right\}},$$

pri čemer se križna korelacija izraža kot množenje Fourierjeve transformacije vhodne slike in kompleksno konjugirane Fourierjeve transformacije predloge. Polje križne korelacije v slikovnem prostoru za $(u,v) = \{u = 1,...,N; v = 1,...,M\}$ nato dobimo z inverzno Fourierjevo transformacijo kot $F^{-1}\left\{\Gamma(\lambda_u, \lambda_v)\right\}$. Podobno lahko pohitrimo tudi izračun ostalih členov v enačbi korelacijskega koeficienta $CC(u,v)$.

4. Napišite funkcijo, ki izračuna polje križne korelacije med vhodno sliko `iImage` in predlogo `iTemplate` z uporabo hitre Fourierjeve transformacije:

   ```
   function oSim = matchCrossCorrFFT( iImage, iTemplate ).
   ```

5. Za izračun hitre Fourierjeve transformacije uporabite ukaza `fft2()` in `ifft2()`, kompleksno konjugiranje pa izvedete z ukazom `conj()`. Pri uporabi hitre Fourierjeve transformacije morate zagotoviti, da bosta vhodna slika in predloga enakih velikosti, zato sliko predloge ustrezno razširite s sivinskimi vrednostmi 0.

   a. Preizkusite delovanje funkcije podobno kot v nalogi (1.a) in primerjajte izhodna polja križnih korelacij `oSim`.

   b. Izmerite čas izvajanja funkcij `matchCrossCorr()` in `matchCrossCorrFFT()` s pomočjo para Matlabovih ukazov `tic`, `toc`. Katera različica je hitrejša in zakaj?

6. Napišite funkcijo, ki izračuna polje korelacijskega koeficienta med vhodno sliko `iImage` in predlogo `iTemplate` z uporabo hitre Fourierjeve transformacije:

   ```
   function oSim = matchCorrCoeffFFT( iImage, iTemplate ).
   ```

7. Izraz v števcu funkcije $CC(u,v)$ lahko poenostavite, če zagotovite da bo $\bar{t} = 0$. Z razčlenitvijo imenovalca funkcije $CC(u,v)$ pa dobite izraze, ki jih hitro izračunate v frekvenčnem prostoru.

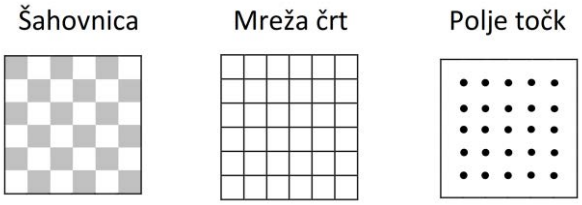
   c. Preizkusite delovanje funkcije podobno kot v nalogi (2.b) in primerjajte izhodna polja korelacijskih koeficientov `oSim`. Kakšen je razpon vrednosti v izhodnih poljih?

   d. Razmislite in obrazložite, kako bi s pomočjo te funkcije in danih predlog načrtali avtomatski postopek za ugotavljanje identitete oseb na danih slikah.

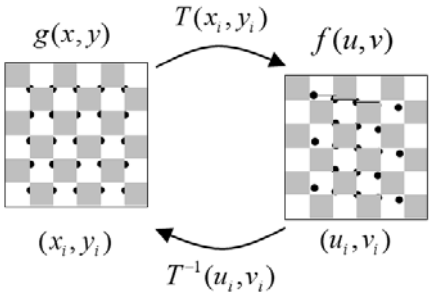
Vaja 7: Geometrijska kalibracija kamere

Kaliber

S postopkom **geometrijske kalibracije** slikovnega sistema zagotovimo enakomerno absolutno velikost slikovnega elementa, ki je sicer v splošnem lahko odvisna od položaja v sliki (projekcija) in dodatno popačena zaradi optičnih aberacij, med katerimi so najpogostejše radialne distorzije tipa sodček in blazinica. Po geometrijski kalibraciji slikovnega sistema lahko enostavno izvajamo meritve dimenzij, ploščin in volumnov objektov na slikah v absolutnih metričnih enotah (npr. mm, mm^2, mm^3). Za geometrijsko kalibracijo 2D slikovnih sistemov uporabljamo ravninske kalibre z enostavnimi, a dobro definiranimi periodičnimi vzorci, kot so šahovnica, mreža tankih črt ali diskretno polje točk.

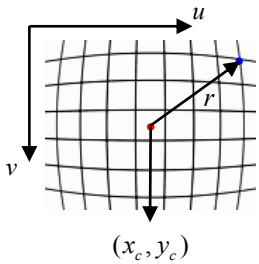


V postopku geometrijske kalibracije poravnamo vzorce zajete $f(u,v)$ in referenčne $g(x,y)$ slike kalibracijskega objekta. V ta namen lahko uporabi celotno slikovno informacijo ali pa izluščimo korespondenčne pare točk referenčne in kalibracijske slike ter z njimi določimo parametre preslikave oz. poravnavo tako, da se preslikane točke referenčne slike čim bolje prekrivajo s korespondenčnimi točkami kalibracijske slike in obratno.



V praksi se za geometrijsko kalibracijo slikovnih sistemov pogosto uporabljata afina ali projektivna geometrijska preslikava, po potrebi pa še ustrezni model za odpravo radialnih distorzij tipa sodček in blazinica. Za ta namen se pogosto uporablja Brownov model radialnih distorzij, ki je določen s centrom distorzij (u_c, v_c) ter parametri radialnih funkcij K_i:

Popačena - zajeta

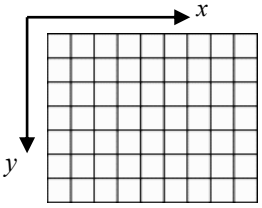


$$x_u = (x_d - x_c)(1 + K_1 r^2 + K_2 r^4 + \ldots)$$
$$y_u = (y_d - y_c)(1 + K_1 r^2 + K_2 r^4 + \ldots)$$
$$r = \sqrt{(x_d - x_c)^2 + (y_d - y_c)^2}$$

(x_c, y_c) - center radialnih distorzij

K_n - parametri radialnih distorzij

Nepopačena – referenčna



Za potrebe geometrijske kalibracije preslikavo izvedemo tako, da izbrane točke na kalibru preslikamo z modelom radialnih distorzij in šele nato z afino oz. projektivno preslikavo.

1. Napišite funkcijo, ki preslika vhodne (x, y) koordinate `iCoorX` in `iCoorY` z Brownovim modelom radialnih distorzij poljubnega reda:

```
function [oCoorX, oCoorY] = transRadialDistortion( ...
            iK, iXc, iYc, iCoorX, iCoorY ),
```

   kjer je `iK` vrstični vektor $1 \times n$ parametrov radialnih distorzij, `iXc` in `iYc` pa prestavljata (x_c, y_c) koordinati centra radialnih distorzij. Funkcija vrne preslikane koordinate točk v spremenljivki `iCoorX` in `iCoorY`.

   a. S pomočjo Matlabove funkcije `ndgrid()` ustvarite diskretno mrežo koordinatnih točk in jih preslikajte s poljubnim Brownovim modelom radialnih distorzij prvega reda. Pri katerih vrednostih parametrov bo preslikava koordinatnih točk identiteta? Obrazložite vpliv položaja centra radialnih distorzij na dobljeno preslikavo koordinatnih točk.

2. Napišite funkcijo, ki preslika vhodne (x, y) koordinate `iCoorX` in `iCoorY` z Brownovim modelom radialnih distorzij poljubnega reda in s poljubno afino ali projektivno preslikavo. Uporabite funkcijo `transRadialDistortion()` in funkciji `transAffine2D()` in `transProjective2D()` (Vaja 5). Pri tem pazite na ustrezni vrstni red preslikav.

   a. Diskretno mrežo koordinatnih točk, ki ste jo ustvarili pri nalogi 1.a uporabite za preslikavo s poljubnim Brownovim modelom radialnih distorzij prvega reda in s poljubno 2D afino ali projektivno preslikavo. Obrazložite vpliv Brownovega modela radialnih distorzij v primeru združenih preslikav.

   b. Ustvarite sliko šahovnice velikosti 100×100 slikovnih elementov s 10×10 polji. Preslikajte sliko šahovnice s preslikavo s poljubnim Brownovim modelom radialnih distorzij prvega reda in s poljubno 2D afino ali projektivno preslikavo.

3. Naložite slike *chess-barrel.tif*, *chess-proj.tif* in *chess-barrel-proj.tif* in jih pretvorite v ustrezne sivinske slike.

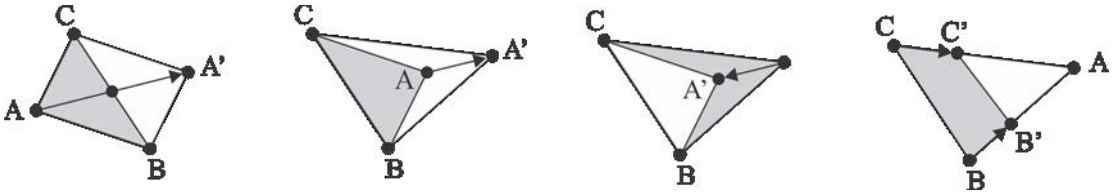
   a. Ročno označite vsaj 8 oglišč na vsaki sliki šahovnice. Pomagajte si s funkcijo `ginput()`.

Izvedba kalibracije

Geometrijsko kalibracijo izvedemo tako, da najprej izbrane točke na kalibru preslikamo z modelom radialnih distorzij in šele nato z afino oz. projektivno preslikavo. Ker je Brownov model radialnih distorzij nelinearen, je za določanje parametrov modela radialnih distorzij potrebno uporabiti optimizacijski postopek, ki minimizira ustrezno kriterijsko funkcijo, npr. povprečno kvadratno Evklidsko razdaljo med korespondenčnimi referenčnimi in izbranimi točkami na kalibru.

Za potrebe določanja parametrov preslikave z modelom radialnih distorij in 2D afine oz. projektivne preslikave boste uporabili Nelder-Mead simpleksni postopek optimizacije. S postopkom simpleksne optimizacije lahko poiščemo lokalni optimum poljubne funkcije več spremeljivk. Simpleks predstavlja geometrijsko strukturo, s pomočjo katere vzorčimo parametrični prostor. Za funkcije dveh spremenljivk $f(x, y)$ je simpleks prestavljen s tremi oglišči oz. s trikotnikom. **Postopek simpleksne minimizacije**: oglišče oz. oglišča z največjo vrednostjo $f(x, y)$ odstranimo in nadomestimo z novimi ogliščem. Novo oglišče oz. oglišča lahko določimo na več načinov:

Zrcaljenje Širjenje Krčenje Mnogokratno krčenje



Vsakič, ko nadomestimo oglišče oz. oglišča dobimo nov simpleks in poiščemo pripadajoče vrednosti $f(x,y)$ v novih ogliščih. Postopek ponavljamo in ga ustavimo, dokler velikost simpleksa ne pade pod neko minimalno vrednost. Položaj lokalnega optimuma bo v oglišču z optimalno vrednostjo $f(x,y)$.

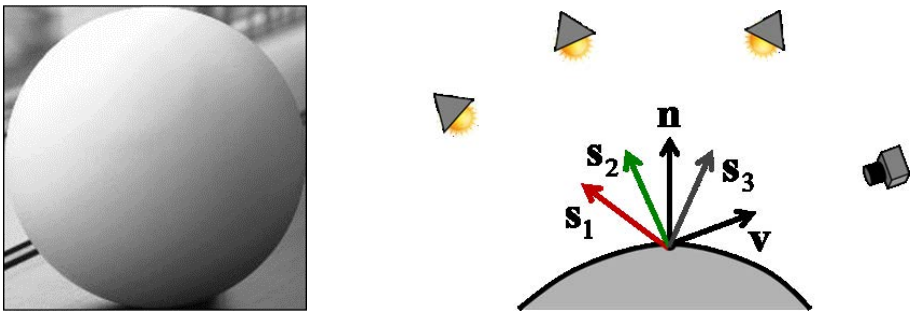
1. Preizkusite delovanje simpleksne optimizacije s pomočjo Matlabove funkcije `fminsearch()`.
    a. Poiščite minimum funkcije $f(x,y) = 100\,(y-x^2)^2 + (1-x)^2$ in uporabite začetni približek $(1.2,1)$. S parametri funkcije `fminsearch()` nastavite prikaz informacij o iteracijah, maksimalno število iteracij 100, najmanjšo napako funkcije 10^{-6} in najmanjši korak 10^{-6}. Navedite minimum funkcije $f(x,y)$ in preverite zaporedje določanje ogliščsimpleksa.
2. Za potrebe geometrijske kalibracije ustvarite referenčno mrežo točk, ki sovpada s položajem ogljišč na idealni, ravninski sliki ogliščšahovnice. Referenčne točke naj ležijo v metričnem prostoru, pri tem pa predpostavite, da ima stranica kvadratnega polja šahovnice dolžino 20 mm. Naprimer, prvo notrjanje oglišče v levem zgornjem kotu šahovnice ima koordinati (20, 20).
3. Izvedite geometrijsko kalibracijo na slikah *chess-barrel.tif*, *chess-proj.tif* in *chess-barrel-proj* s pomočjo korespondenčnih točk, ki ste jih določili pri prejšnji vaji.
    a. Izračunajte parametre aproksimacijske afine preslikave, ki vam poravnajo referenčne točke s korespondenčnimi točkami zaznanih ogliščPrikažite referenčne točke in točke ogliščpred in po preslikavi z izračunano aproksimacijsko afino preslikavo. Ali je afina preslikava primerna za geometrijsko kalibracijo slike?
    b. Določite parametre aproksimacijske projektivne preslikave tako, da uporabite simpleksno minimizacijo povprečne kvadratne Evklidske razdalje med referenčnimi točkami in korespondenčnimi točkami zaznanih ogliščKako ste izbrali začetne vrednosti parametrov projektivne aproksimacijske preslikave?
    c. Geometrijsko kalibracijo izvedite z uporabo Brownovega modela radialnih distorzij (prvi red) tako, da uporabite simpleksno minimizacijo povprečne kvadratne Evklidske razdalje med referenčnimi točkami in korespondenčnimi točkami zaznanih ogliščZačetne vrednosti parametrov radialne transformacije *Ki* postavite na 0, center radialnih distorzij pa v geometrično središče slike.
    d. Poleg Brownovega modela radialnih distorzij (prvi red) uporabite še projektivno preslikavo v postopku simpleksne minimizacije povprečne kvadratne Evklidske razdalje med referenčnimi točkami in korespondenčnimi točkami zaznanih ogliščParametri optimizacije naj obsegajo le parametre modela radialnih distorzij, parametre aproksimativne projektivne preslikave pa določite v vsaki iteraciji posebej. S katero geometrijsko preslikavo ste dobili najmanjšo končno povprečno kvadratno Evklidsko razdaljo?

4. Predpostavite, da ima stranica kvadratnega polja šahovnice dolžino 20 mm. Preslikajte slike šahovnice z določenimi geometrijskimi preslikavami tako, da bodo na izhodnih, kalibriranih slikah velikosti slikovnih elementov 1/5 mm.
    a. Prikažite slike šahovnice pred in po geometrijskih preslikavah.
5. Postopek kalibracije ponovite tako, da večkrat zaporedoma ročno označite izbrane točke oglišč na sliki šahovnice. Na podlagi dobljenih rezultatov ocenite točnost celotnega postopka geometrijske kalibracije.
    a. Od česa zavisi točnost geometrijske kalibracije?
6. Razvijte postopek za avtomatsko geometrijsko kalibracijo. S pomočjo funkcij za robustno iskanje 2D objektov (Vaja 4) določite koordinate ogljišč šahovnice. Določite ustrezno mrežo referenčnih točk in določite korespondence z zaznanimi točkami oglišč na slikah *chess-barrel.tif*, *chess-proj.tif* in *chess-barrel-proj*. Korespondence določite s postopkom ICP (Vaja 5).
    a. Na kakšen način bi morali spremeniti postopek določanja korespondenc ICP za uporabo z 2D afino oz. projektivno preslikavo?
    b. Geometrijsko kalibracijo izvedite z Brownovim modelom radialnih distorzij (prvi red) in projektivno preslikavo. Kolikšna je točnost avtomatske v primerjavi z ročno geometrijsko kalibracijo?

Vaja 8: Rekonstrukcija 3D oblik

Fotometrični stereo

Pri rekonstrukciji 3D oblik želimo s pomočjo ene ali več 2D slik nekega objekta, zajetih iz različnih pogledov in/ali pod različnimi osvetlitvami, najti optimalno 3D obliko tega objekta tako, da bo 3D oblika čim bolje sovpadala z zajetimi 2D slikami. Eden od postopkov rekonstrukcije 3D oblik je **fotometrični stereo**, pri katerem izluščimo informacijo o obliki na podlagi senc na objektu. Sence ustvarimo z osvetlitvijo objekta iz različnih smeri ($\mathbf{s}_1, \mathbf{s}_2, \mathbf{s}_3$), slike pa zajamemo iz enega pogleda ($\mathbf{v}$).



Če predpostavimo, da pri interakciji svetloba-površina pride le do difuznega odboja, potem lahko enostavno povežemo svetilnost slikovnega elementa z obliko površine. Uporabimo **Lambertov model difuzne površine**:

$$I(x_i, y_i) = \frac{\rho}{\pi} kc \cos\theta_i = \frac{\rho}{\pi} kc\, \mathbf{n}_i \cdot \mathbf{s}$$

kjer je *k* svetilnost vira, ρ albedo oz. odbojnost površine in *c* konstanta optičnega sistema. Če predpostavimo, da je $\frac{\rho}{\pi} kc = 1$ potem velja:

Lambertov model: $I(x_i, y_i) = \cos\theta_i = \mathbf{n}_i \cdot \mathbf{s}$ ⟶ Sistem enačb za tri svetila: $I_1 = \rho\mathbf{n}\cdot\mathbf{s}_1$, $I_2 = \rho\mathbf{n}\cdot\mathbf{s}_2$, $I_3 = \rho\mathbf{n}\cdot\mathbf{s}_3$ ⟶ $\begin{bmatrix} I_1 \\ I_2 \\ I_2 \end{bmatrix} = \begin{bmatrix} \mathbf{s}_1^T \\ \mathbf{s}_2^T \\ \mathbf{s}_3^T \end{bmatrix} \rho\mathbf{n}$ ($\mathbf{I}_{(3\times1)}$, $\mathbf{S}_{(3\times3)}$, $\tilde{\mathbf{n}}_{(3\times1)}$) ⟶ Rešitev: $\tilde{\mathbf{n}} = \mathbf{S}^{-1}\mathbf{I}$, $\rho = \|\tilde{\mathbf{n}}\|$, $\mathbf{n} = \frac{\tilde{\mathbf{n}}}{\|\tilde{\mathbf{n}}\|} = \frac{\tilde{\mathbf{n}}}{\rho}$

Z gornjim postopkom dobimo oceno normale $\mathbf{n}$ na površino objekta v vsaki točki slike. Ocena normale $\mathbf{n}$ je bolj robustna, če uporabimo več osvetlitev objekta iz več med seboj različnih smeri. V posameznih pogledih so lahko nekateri slikovni elementi relativno temni, zato bo ocena normale tam manj zanesljiva. To rešimo z uteževanjem posameznih prispevkov s svetlostjo $I(x_i, y_i)$. Dobimo naslednji sistem enačb:

$$I_1(I_1) = I_1(\rho\mathbf{n}\cdot\mathbf{s}_i) \quad \vdots \quad I_N(I_N) = I_N(\rho\mathbf{n}\cdot\mathbf{s}_N) \longrightarrow \underbrace{\begin{bmatrix} I_1^2 \\ \vdots \\ I_N^2 \end{bmatrix}}_{\mathbf{I}_{(N\times 1)}} = \underbrace{\begin{bmatrix} I_1\mathbf{s}_1^T \\ \vdots \\ I_N\mathbf{s}_N^T \end{bmatrix}}_{\mathbf{S}_{(N\times 3)}} \underbrace{\rho\mathbf{n}}_{\tilde{\mathbf{n}}_{(3\times 1)}} \longrightarrow$$

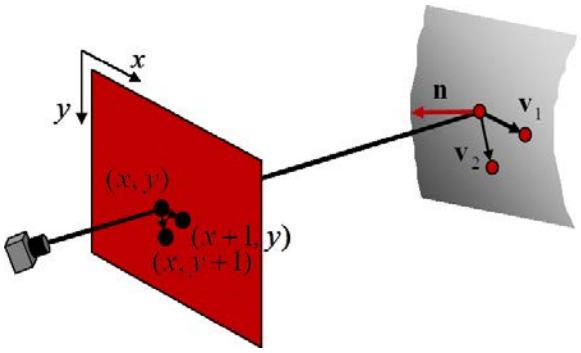
Rešitev:

$$\mathbf{I} = \mathbf{S}\tilde{\mathbf{n}}$$
$$\mathbf{S}^T\mathbf{I} = \mathbf{S}^T\mathbf{S}\tilde{\mathbf{n}}$$
$$\tilde{\mathbf{n}} = \left(\mathbf{S}^T\mathbf{S}\right)^{-1}\mathbf{S}^T\mathbf{I}$$

S sliko normal na površino lahko rekonstruiramo 3D obliko objekta na več načinov. Najenostavnejši način je, da integriramo sliko normal $\mathbf{n} = (n_x\ n_y\ n_z)^{\mathrm{T}}$ po poljubni krivulji do izbrane točke (x, y):

$$z(x,y) = \int_0^x \frac{n_x(u,y)}{n_z(u,y)}\,du + \int_0^y \frac{n_y(x,v)}{n_z(x,v)}\,dv$$

Razdalje oz. globino $z(x,y)$ lahko določimo tudi z analizo $(\partial x, \partial y)$ odvodov površine, ki morajo biti v vsaki točki slike pravokotni na normalo $\mathbf{n}$:



$$\mathbf{v}_1 = (x+1, y, z(x+1,y)) - (x,y,z(x,y)) = (1, 0, z(x+1,y) - z(x,y))$$
$$\mathbf{v}_2 = (x, y+1, z(x,y+1)) - (x,y,z(x,y)) = (0, 1, z(x,y+1) - z(x,y))$$

Velja:

$$\mathbf{n}\cdot\mathbf{v}_1 = 0,\ \mathbf{n}\cdot\mathbf{v}_2 = 0$$

Rešitev:

$\mathbf{Mz} = \mathbf{v} \rightarrow \mathbf{M}$ je redka matrika

$$\mathbf{z} = \left(\mathbf{M}^{\mathrm{T}}\mathbf{M}\right)^{-1}\mathbf{M}^{\mathrm{T}}\mathbf{v}$$

Naloge

Za izvedbo vaje boste naložili datoteko *owl.zip*, ki vsebuje datoteko *lights.txt*, v kateri so podane smeri svetil glede na objekt, slike objekta *owl.**X**.tif*, ***X***=0,…,11 in pripadajočo masko objekta *owl.mask.tif*.

1. Napišite funkcijo, ki izračuna normale $\mathbf{n}$ na površino objekta iz poljubnega števila slik objekta:

```
function oNormals = computeNormals( iImages, iMask,
                                    iLightDir )
```

   kjer je `iImages` matrika dimezij $U \times V \times N$ (U, V sta dimenziji 2D slike, N pa število slik), `iMask` je maska objekta in `iLightDir` matrika $3 \times N$ s smermi osvetlitve posamezne slike v matriki `iImages`. Funkcija vrne matriko normal dimenzij $U \times V \times 3$ v spremenljivki `oNormals`.

   a. Preizkusite delovanje funkcije s tremi vhodnimi slikami, pri tem pa pozorno izberite vhodno sliko glede na dane smeri osvetlitve. Prikažite sliko normal po komponentah s funkcijo `imagesc()` in v obliki vektorskega polja s funkcijo `quiver()`. Ali so smeri normal smiselne glede na dano sliko? Obrazložite vpliv izbire vhodnih slik na rezultat.

   b. Preizkusite delovanje funkcije z vsemi 12 vhodnimi slikami. Obrazložite rezultat v primerjavi s prejšnjimi rezultati na treh vhodnih slikah.

2. Napišite funkcijo, ki izračuna albedo oz. odbojnost površine objekta s pomočjo normal slike:

   ```
   function oAlbedo = computeAlbedo( iNormals, iMask ),
   ```

   kjer je `iNormals` matrika dimezij $U \times V \times 3$ (U, V sta dimenziji 2D slike, vsaka normala pa ima tri komponente), `iMask` je maska objekta. Pazite, da bodo normale v nenormalizirani obliki $\tilde{\mathbf{n}}$. Funkcija vrne matriko albedov z dimenzijami $U \times V$ v spremenljivki `oAlbedo`. Zagotovite, da bodo vrednosti v `oAlbedo` na območju [0, 1].

   a. Preizkusite delovanje funkcije za slike normal, ki ste jih izračunali pri nalogah (1.a) in (1.b). Obrazložite zakaj se razlikujejo vrednosti albeda glede na položaj na objektu.

   b. Prikažite slike albeda za posamezne kanale RGB slike in obrazložite razlike med njimi.

3. Napišite funkcijo, ki rekonstruira obliko objekta oz. vsaki točki na objektu pripiše višino $z(x, y)$:

   ```
   function oDepth = computeDepth( iNormals, iMask ),
   ```

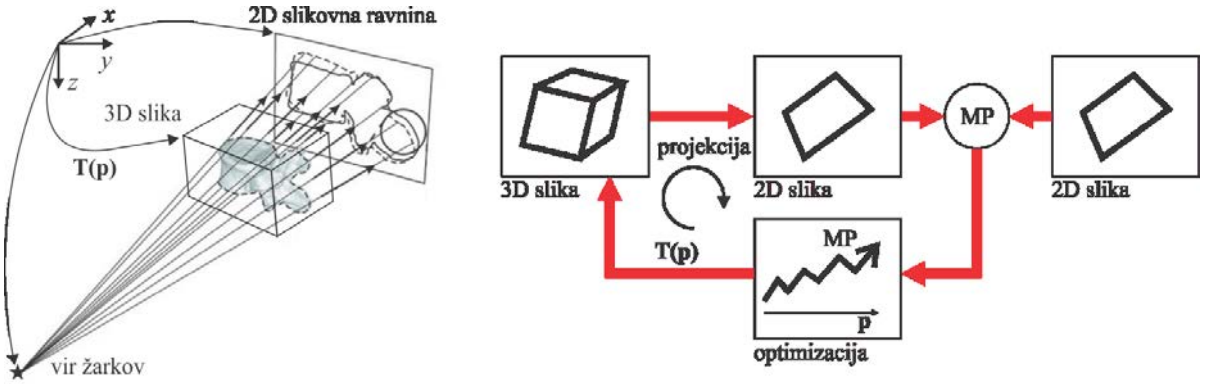
   kjer je `iNormals` matrika dimezij $U \times V \times 3$, `iMask` je maska objekta. Pazite, da bodo normale v normalizirani obliki $\mathbf{n}$. Funkcija vrne matriko z višinami $z(x, y)$ dimenzij $U \times V$ v spremenljivki `oDepth`. Če boste uporabili rekonstrukcijo z reševanjem sistema $\mathbf{z} = (\mathbf{M}^{\mathrm{T}}\mathbf{M})^{-1}\mathbf{M}^{\mathrm{T}}\mathbf{v}$, potem matriko M (dimenzije $2UV \times UV$) inicializirajte z ukazom `sparse()`. Za reševanje sistema uporabite Matlabov operatov `\`, naprimer z ukazom `z=M\v`.

   a. Preizkusite delovanje funkcije za slike normal, ki ste jih izračunali pri nalogah (1.a) in (1.b). Prikažite rekonstruirane površine s funkcijo `surf()`. Kje pride do največjih razlik med rekonstrukcijami in zakaj?

Vaja 9: Prileganje 3D modelov na 2D slike

Zasnova algoritma

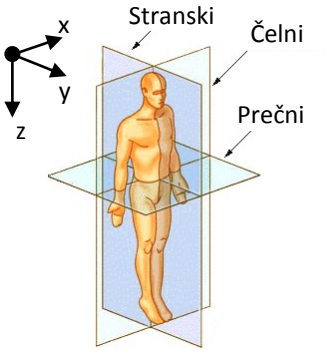
Prileganje 3D modelov ali slik objekta zanimanja na 2D slike omogoča določanje položaja teh objektov v 3D prostoru. Prileganje 3D modelov na 2D slike izvedemo s postopki **3D-2D poravnave**. 3D-2D poravnava je proces iskanja optimalnih geometrijskih preslikav T, ki nam 3D model oz. sliko preslikajo tako, da bo le-ta pri dani geometrijski postavitvi sovpadala s slikovno informacijo na 2D slikah. Ključen problem pri 3D-2D poravnavi slik je prostorska neskladnost informacije (3D vs. 2D), ki jo lahko rešimo ali **s projekcijo 3D informacije v 2D slikovni prostor** ali z rekonstrukcijo 3D infomacije na podlagi ene ali večih 2D slik. Prostorsko ujemanje informacije nato lahko ovrednotimo z mero podobnosti (MP), in sicer s primerjavo informacije v 2D oz. v 3D, odvisno ali uporabimo projekcijo oz. rekonstrukcijo. Tekom postopka 3D-2D poravnave nam izbrana optimizacijska metoda iterativno spreminja parametre geometrijske preslikave $p \rightarrow T(p)$, in sicer tako, da teži k optimalni vrednosti mere podobnosti MP. Ena od možnih izvedb 3D-2D poravnave s projekcijo iz 3D v 2D je prikazana na spodnji sliki, pri kateri želimo maksimizirati ujemanje med 2D sliko in projicirano 2D sliko.

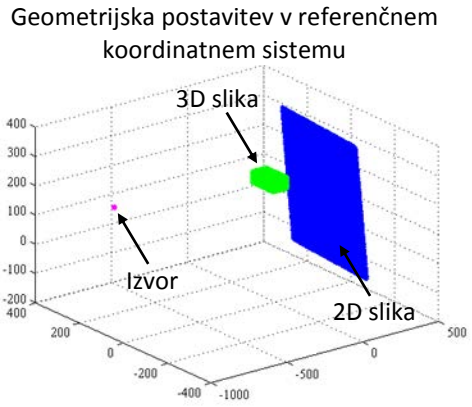


Pri vaji boste načrtali 3D-2D poravnavo s projekcijo iz 3D v 2D. Način projekcije informacije iz 3D v 2D prostor zavisi od predvsem od oblike 3D informacije. Pri vaji boste simulirali delovanje rentgena, pri katerem se projekcija iz 3D v 2D izraža kot integral po premici od izvora rentgenskih žarkov do 2D slikovne ravnine. Tovrstnim projekcijam pravimo tudi digitalno rekonstruirani rentgenski posnetek oz. DRR (ang. *Digitally Reconstructed Radiographs*) . Kot 3D model boste uporabili sliko zajeto z računalniško tomografijo (ang. *Computed Tomography*). Za potrebe 3D-2D poravnave se pogosto uporablja tudi projekcija maksimalne vrednosti MIP (ang. *Maximum Intensity Projection*).

V datoteki *vretenca.mat* boste našli strukturi `ct` in `Xray`, ki podajata 3D sliko ledvenega vretenca L3 zajeto z računalniško tomografijo in 2D rentgensko sliko ledvenih vretenc L1-L5. V strukturah `ct` in `Xray` so dane matrike sivinskih vrednosti (`ct.volume` oz. `Xray.image`), geometrijske preslikave TPos koordinatnih sistemov 3D in 2D slik iz referenčnega koordinatnega sistema v prvi slikovni element slik z indeksom (1,1) ter položaj izvora rentgenskih žarkov SPos v referenčnem koordinatnem sistemu.

1. Naložite 2D rentgensko sliko `Xray.image`. Sivinske vrednosti slike preslikajte z linearnim oknjenjem iz območja [0,120] v območje [0,255] in sliko prikažite.
    a. Koliko vretenc je vidnih na 2D sliki?
2. Naložite in prikažite 3D CT sliko `ct.volume` v stranskem, čelnem in prečnem 2D pogledu.





3. Ustvarite vzorčni mreži 2D rentgenske in 3D CT slike s pomočjo Matlabovega ukaza `ndgrid()`. Točke na vzorčnih mrežah preslikajte v referenčni koordinatni sistem s pripadajočimi preslikavami `TPos`. Prikažite preslikane točke vzorčnih mrež in položaj izvora rentgenskih žarkov `SPos` s pomočjo ukaza `plot3()`. Preverite pravilnost dobljene geometrijske postavitve izvora, 3D in 2D slike v refenčnem koordinatnem sistemu s pomočjo gornje desne slike.
4. Napišite funkcijo, ki preslika poljubno točko `iPos` v 3D prostoru na 2D slikovno ravnino:

```
function oPos = mapPoint2Plane( iPos, Xray ),
```

kjer je `iPos` stolpni vektor 3×1, `Xray` pa struktura s podatki o izvoru rentgenskih žarkov in 2D slikovni ravnini. Funkcija vrne točko `oPos` v obliki stolpnega vektorja 3×1, ki ga določite kot presečišče 2D slikovne ravnine s premico, ki je določena s točkama `SPos` in `iPos`.

   a. S pomočjo funkcije preslikajte koordinate oglišč 3D slike na 2D slikovno ravnino in točke vrišite v celotno geometrijsko postavitev kot pri nalogi (3).

   b. Poiščite najmanjši možni pravokotnik v 2D rentgenski sliki, ki vsebuje vseh 8 preslikanih koordinat oglišč 3D slike. Točke znotraj pravokotnika ustrezno označite oz. vrišite v celotno geometrijsko postavitev kot pri nalogi (3).

5. Napišite funkcijo za projekcijo 3D CT slike v 2D slikovno ravnino:

```
function [oImage, oMask] = project3D2D( ct, Xray, iStep ),
```

kjer je `ct` struktura s podatki o 3D CT sliki, `Xray` struktura s podatki o izvoru rentgenskih žarkov in 2D slikovni ravnini, `iStep` pa korak vzorčenja vzdolž premic od izvora rentgenskih žarkov do 2D slikovne ravnine. Funkcija vrne 2D sliko `oImage` dimenzij $U\times V$ (enako velikosti 2D slike `Xray.image`) in 2D sliko `oMask` z najmanjšim pravokotnim področjem, ki vsebuje področje projicirane 3D slike v `oImage` - naloga (4.b). Projicirano 3D sliko ustvarite tako, da najprej določite najmanjše pravokotno področje v 2D slikovni ravnini, kamor se bo projicirala 3D slika. Vsako točko v tem področju povežite z izvorom rentgenskih žarkov s premico in nato določite vzorčne točke na premici s korakom `iStep` od izvora do 2D slikovne ravnine. Sivinske vrednosti 3D slike v vzorčnih točkah določite s trilinearno interpolacijo z Matlabovo funkcijo `interpn()`. Projicirano sivinsko vrednost izračunate kot povprečje (DRR) oz. maksimum (MIP) vzorčenih sivinskih vrednosti. *Namig:* vzorčite le tiste točke na premicah, ki ležijo znotraj 3D slike.

   a. Preizkusite delovanje funkcije s pomočjo danih slik in ustvarite DRR in MIP sliki. Obrazložite vpliv izbire koraka `iStep` na projicirani sliki.

b. Razširite funkcijo `project3D2D()` tako, da bo imela dodatni vhodni parameter `iRigid3D`. Parameter `iRigid3D` je homogena matrika 4×4, ki predstavlja poljubno togo preslikavo v 3D in ki jo določite s funkcijo `transAffine3D()` (Vaja 5). S `iRigid3D` izvedite geometrijsko preslikavo 3D CT slike predno izvedete projekcijo v 2D. Preizkusite delovanje funkcije s poljubno togo preslikavo.

Mera podobnosti in optimizacija

Mera podobnosti je poljubna skalarna funkcija, določena nad vsemi istoležnimi slikovnimi elementi referenčne slike $a(x,y)$ in lebdeče slike $b(x,y)$, ki ima optimalno vrednost pri optimalni poravnavi slik. 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 slik se pogosto uporabljata naslednji dve meri podobnosti:

- **Korelacijski koeficient** CC (ang. *Correlation Coefficient*):

$$CC(a,b)=\frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(a(x_i,y_j)-\bar{a}\right)\left(b(x_i,y_j)-\bar{b}\right)}{\sqrt{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(a(x_i,y_j)-\bar{a}\right)^2\cdot\sum_{i=1}^{I}\sum_{j=1}^{J}\left(b(x_i,y_j)-\bar{b}\right)^2}}.$$

- **Medsebojna informacija** MI (ang. *Mutual Information*):

$$MI(a,b)=H(a)+H(b)-H(a,b),$$

Kjer je $H(a)$ entropija referenčne slike $a(x,y)$, $H(b)$ entropija lebdeče slike $b(x,y)$, $H(a,b)$ pa njuna skupna entropija:

$$\begin{aligned}H(a)&=-\sum_{s_a=0}^{L-1}p_a(s_a)\cdot\log(p_a(s_a))\\H(b)&=-\sum_{s_b=0}^{L-1}p_b(s_b)\cdot\log(p_b(s_b))\\H(a,b)&=-\sum_{s_a=0}^{L-1}\sum_{s_b=0}^{L-1}p_{ab}(s_a,s_b)\cdot\log(p_{ab}(s_a,s_b))\end{aligned},$$

Verjetnostni porazdelitvi $p_a(s_a)$ in $p_b(s_b)$ ter skupno porazdelitev $p_{ab}(s_a,s_b)$ ocenimo z normalizacijo pripadajočih histogramov $h_a(s_a)$, $h_b(s_b)$ ter $h_{ab}(s_a,s_b)$:

$$p_a(s_a)=\frac{h_a(s_a)}{I\cdot J},\qquad p_a(s_a)=\frac{h_b(s_b)}{I\cdot J},\qquad p_{ab}(s_a,s_b)=\frac{h_{ab}(s_a,s_b)}{I\cdot J}$$

Spremenljivki s_a in s_b označujeta diskretne sivinske vrednosti referenčne slike $a(x,y)$ in lebdeče slike $b(x,y)$.

1. Napišite funkcijo, ki izračuna vrednost korelacijskega koeficienta med dvema slikama:

```
function oCC = correlationCoefficient( iImageA, iImageB ),
```

kjer sta `iImageA` in `iImageB` sliki dimezij $U\times V$. Funkcija vrne vrednost korelacijskega koeficienta `oCC`.

a. Preizkusite delovanje funkcije tako, da s pomočjo poljubnih togih preslikav 3D CT slike ustvarite različne 2D projekcije in izračunate `oCC` med projicirano in rentgensko sliko. Kakšen je maksimalen razpon vrednosti korelacijskega koeficienta?

b. Izračunajte in izrišite vrednosti `oCC` med projiciranimi in rentgensko sliko tako, da spreminjate le po en parameter toge preslikave $\boldsymbol{p}=[t_x,\ t_y,\ t_z,\ \varphi_x,\ \varphi_y,\ \varphi_z]^{\mathrm{T}}$. Naprimer, translacije t spreminjajte od -20 do 20 s korakom 2, rotacijo φ pa od -10 do 10 s korakom 1. Obrazložite potek korelacijskega koeficienta. Ali vrednost korelacijskega koeficienta odraža dejansko podobnost med slikami?

2. Napišite funkcijo, izračuna vrednost medsebojne informacije med dvema slikama:

```
function oMI = mutualInformation( iImageA, iImageB ),
```

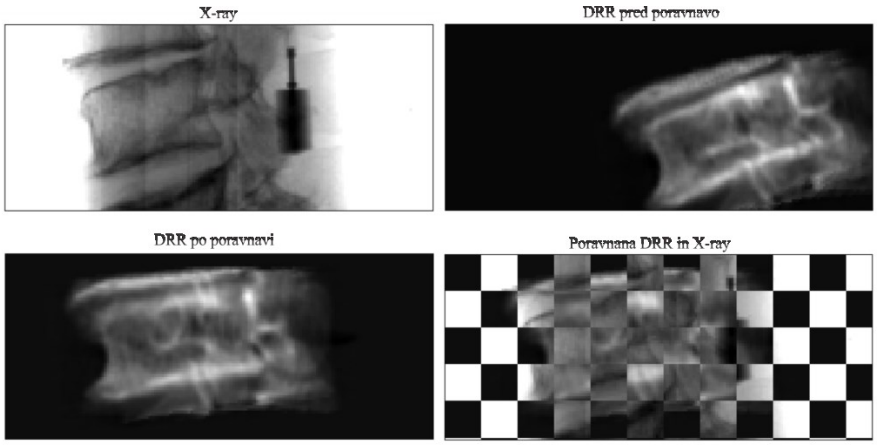
kjer sta `iImageA` in `iImageB` sliki dimezij $U \times V$. Funkcija vrne vrednost medsebojne informacije `oMI`.

a. Preizkusite delovanje funkcije tako, da s pomočjo poljubnih togih preslikav 3D CT slike ustvarite različne 2D projekcije in izračunate `oMI` med projicirano in rentgensko sliko. Kakšen je maksimalen razpon vrednosti medsebojne informacije?

b. Izračunajte in izrišite vrednosti `oMI` med projiciranimi in rentgensko sliko tako, da spreminjate le po en parameter toge preslikave $\boldsymbol{p}=[t_x,\ t_y,\ t_z,\ \varphi_x,\ \varphi_y,\ \varphi_z]^{\mathrm{T}}$. Naprimer, translacije t spreminjajte od -20 do 20 s korakom 2, rotacijo φ pa od -10 do 10 s korakom 1. Obrazložite potek medsebojne informacije.

3. Načrtajte avtomatski postopek za togo 3D-2D poravnavo. Ustvarite kriterijsko funkcijo, ki pri danih parametrih toge preslikave 3D CT slike le-to projicira v 2D slikovno ravnino in izračuna ter vrne mero podobnosti med projicirano in rentgensko sliko. S pomočjo simpleksne optimizacije z Matlabovo funkcijo `fminsearch()` poiščite optimalne parametre toge preslikave 3D CT slike.

a. Preverite delovanje funkcije z uporabo slik `ct` in `ct2` ter preverite prostorsko ujemanje po poravnavi podobno kot je prikazano na spodnji sliki.

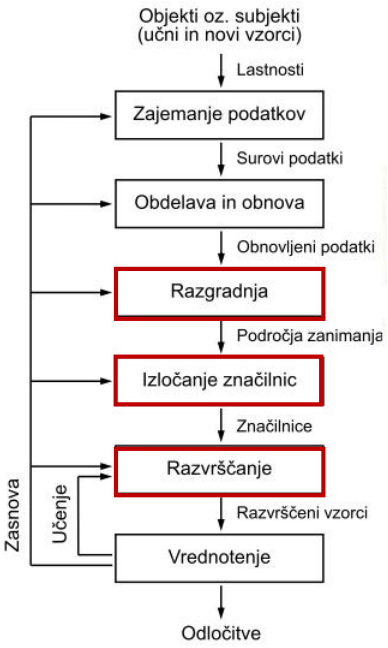


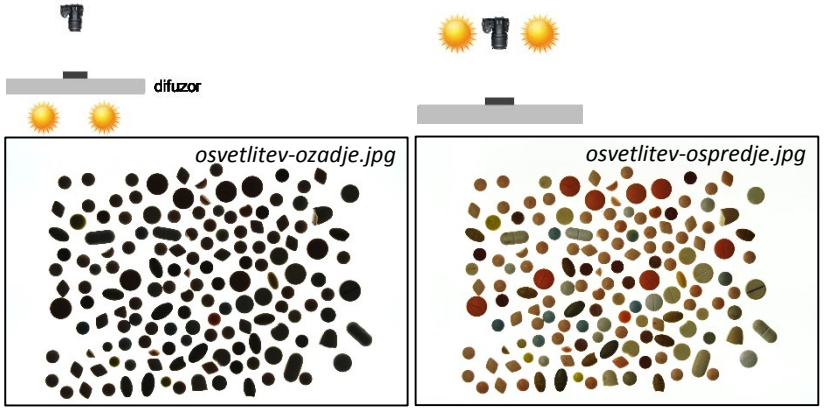
Vaja 10: Vizualna kontrola kakovosti

Zasnova sistema odločanja na podlagi slike

Vizualna kontrola kakovosti je proces razpoznavanja vzorcev na podlagi vizualne oz. slikovne informacije, ki je namenjen podpori pri odločanju o ustreznosti oz. neustreznosti objektov zanimanja. Interpretacija surovih slikovnih podatkov je pogosto zelo zamudna, naporna in včasih tudi zahtevna naloga, zato je smiselno razvijati računalniške postopke za podporo pri odločanju, ki temeljijo na

avtomatski analizi, opisovanju in razpoznavanju opazovanih objektov. Splošni sistem za razpoznavanje vzorcev lahko obravnavamo kot zaporedje šestih osnovnih funkcionalnih podsistemov: zajemanje podatkov, obdelava in obnova zajetih podatkov, razgradnja obnovljenih podatkov, izločanje značilnic, razvrščanje ter vrednotenje razvrščanja. Pri vaji se bomo spoznali z osnovno razgradnjo slik, izločanjem značilnic in razvrščanjem za problem vizualne kontrole kakovosti tablet. Kratek opis problema je podan spodaj.





Problem: na podlagi gornjih dveh slik želimo načrtati vizualno kontrolo kakovosti tablet, ki zazna in označi kot ustrezne le okrogle tablete z zarezo, ostale pa kot neustrezne. Ključna vprašanja pri izvedbi vizualne kontrole kakovosti so:

- **Kje na sliki se nahajajo posamezne tablete?**
- **Katere tablete so okrogle?**
- **Katere okrogle tablete imajo zarezo?**

Razgradnja slik

Razgradnja ali segmentacija slik (ang. image segmentation) združuje postopke, s katerimi sliko razdelimo na osnovna področja oziroma objekte. Razgradnjo bomo izvedli z upragovljanjem slike z osvetlitvijo iz ozadja in z označevanjem objektov. Pri označevanju objektov vsakemu (binarnemu) objektu na sliki priredimo lastno oznako oz. kodo.

1. Naložite sliko *osvetlitev-ozadje.jpg* in jo z upragovljanjem pretvorite v binarno sliko tako, da določite sivinsko vrednost praga, ki področje tablet loči od ozadja. S pomočjo Matlabove funkcije `bwlabel()` označite objekte v binarni sliki.

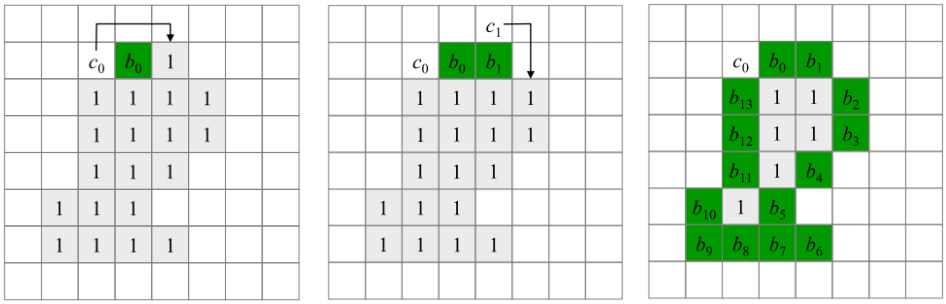
    b. Koliko objektov oz. tablet je na 2D sliki?

Analiza objektov zanimanja in binarni razvščevalnik

Analizo objektov v razgrajeni sliki izvedemo z izločanjem značilnic oz. kvantitativnimi opisi lastnosti posameznih področij slike. Posamezno področje slike lahko predstavimo **z zunanjimi lastnostmi** (mejami področja) in/ali **notranjimi lastnostmi** (slikovnimi elementi področja). Zunanje lastnosti omogočajo predstavitev in opisovanje področij z izločanjem značilnic iz mej področij, npr. obseg, usmeritev, največja dolžina, gladkost meje, itd. Notranje lastnosti po drugi strani omogočajo predstavitev in opisovanje pojavnosti področij, kot so npr. homogenost, barva, tekstura, itd.

Meje področja v binarni sliki (0 – ozadje, 1 - objekt) lahko določimo s Moorovim postopkom sledenja mej. Mejo predstavimo kot zaporedje mejnih točk v smeri urinega kazalca:

1. Določimo začetno točko b_0 kot najvišjo in najbolj levo točko z vrednostjo 1 ter označimo s c_0 sosednjo točko na levi, ki ima vrednost 0. Preiščemo 8 sosednjih točk začetne točke b_0, tako da začnemo v točki c_0 in nadaljujemo v smeri urinega kazalca. Z b_1 označimo prvo sosednjo točko z vrednostjo 1, s c_1 pa točko z vrednostjo 0 tik pred b_1 v zaporedju sosednjih točk. Zapomnimo si b_0 in b_1.
2. Naj bo $b=b_1$ in $c=c_1$.
3. Označimo 8 sosednjih točk $n_1,n_2,\ldots,n_8$ točke b, ki se začnejo v točki c in se nadaljujejo v smeri urinega kazalca, ter poiščemo prvo točko n_k z vrednostjo 1.
4. Naj bo $b=n_k$ in $c=n_{k-1}$.
5. Ponavljamo koraka 3 in 4 dokler ne pridemo do začetne točke $b=b_0$ in za njo najdemo isto naslednjo točko b_1.



Po končanem postopku zaporedje točk b_i predstavlja zaporedje zunanjih mejnih točk objekta v smeri urinega kazalca.

Najosnovnejši opis **zunanje lastnosti področja** je njegova površina A, ki je določena s številom slikovnih elementov področja. Obseg P področja je enak dolžini njegove meje. Opis oblike področja, ki je neodvisen od velikosti in orientacije objekta je krožno razmerje:

$$K = \frac{4\pi A}{P^2},$$

ki ima vrednost 1 za področja v obliki kroga, sicer pa je manjše od 1.

Notranje lastnosti področja najbolj preprosto opišemo s statističnimi momenti sivinskih vrednosti področij slike. Naj bo $p(z)$ verjetnostna porazdelitev sivinskih vrednosti znotraj posameznih področij, ki jo ocenimo z normalizacijo histograma sivinskih vrednosti znotraj posameznih področij slike. Centralni moment n-te stopnje izračunamo kot:

$$\mu_n = \sum_{i=0}^{L-1} (z_i - m)^n \, p(z_i)$$, kjer povprečje m izračunamo kot $m = \sum_{i=0}^{L-1} z_i \, p(z_i)$.

Ničti moment μ_0 je vedno enak 1, prvi moment μ_1 pa 0. Drugi moment $\mu_2 = \sigma^2$ predstavlja varianco, ki je značilnica razpršenosti oz. kontrasta sivinskih vrednosti, in ga lahko uporabimo za opis gladkosti področja:

$$G = \frac{1}{1+\sigma^2}.$$

Ta ima vrednost 0 na popolnoma homogenih področjih in se približuje 1 na področjih z veliko varianco.

1. Napišite funkcijo, ki določi koordinate meje posameznega področja na 2D sliki:

```
function oContour = trackContour( iLabel, iObject ),
```

kjer je `iLabel` slika označenih objektov, `iObject` pa je oznaka objekta, katerega mejo želimo določiti. Funkcija vrne spremenljivko `oContour` v obliki matrike $2\times M$ z zaporedjem koordinat meje $[x_i \; y_i]^{\mathrm{T}}, i = 1,\ldots,M$.

a. Določite meje vseh označenih objektov iz naloge 1. Naložite in prikažite sliko *osvetlitev-ospredje.jpg* ter v sliko vrišite meje objektov.

b. Izračunajte obseg oz. dolžino meje označenih objektov in narišite histogram vseh izračunanih obsegov.

2. Napišite funkcijo, ki izloči značilnice posameznega področja na 2D sliki:

```
function oFeatures = extractFeatures( iImage, iLabel,
                                      iObject )
```

kjer je `iImage` sivinska slika z osvetlitvijo iz ospredja, `iLabel` slika označenih objektov, `iObject` pa je oznaka objekta, katerega značilnice želimo izračunati. Funkcija vrne spremenljivko `oFeatures` v obliki vektorja z značilnicami $[A, P, K, \mu_2, G]^{\mathrm{T}}$.

a. Narišite histograme posameznih značilnic za vse objekte v sliki. Razmislite katera oz. katere od značilnic bi bile primerne za zaznavo okroglih tablet in tablet z zarezo?

b. Določite vrednosti pragov izbranih značilnic tako, da boste pravilno zaznali čim večje število okroglih tablet z zarezo. Razmislite na kakšen način bi preverili zanesljivost delovanja tega procesa razvrščanja.