Univerza v Ljubljani, Fakulteta za elektrotehniko
Podiplomski magistrski študijski program 2. stopnje Elektrotehnika
Informacijsko komunikacijske tehnologije - Obdelava slik in videa (64238)

NAVODILA

Vaja 9: Geometrijska poravnava slik

Pripravili: Luka Škrlj, Gašper Podobnik & Tomaž Vrtovec

Navodila

Naslednje naloge se opravi med laboratorijsko vajo.

Poravnava *kontrolnih točk* temelji na iskanju geometrijske preslikave $\mathcal{T}$, ki določa najmanjšo razdaljo med preslikanimi točkami $\mathcal{T}(x_k, y_k)$; $k = 1, 2, \ldots, K$, na referenčni sliki $a(x, y)$ in korespondenčnimi točkami (u_k, v_k); $k = 1, 2, \ldots, K$, na vhodni sliki $b(u, v)$.

1. Dana je arhivska datoteka vaja09.zip v ZIP formatu. Datoteke, ki se nahajajo v arhivu, shranite v izbrano mapo ter v Pythonu poženite uporabniški vmesnik vaja09.py (*desni klik*, *Run Python File in Terminal*). Uporabniški vmesnik omogoča nalaganje referenčne in vhodne slike, izbiro kontrolnih točk, izvedbo izbrane poravnave ter prikaz rezultatov poravnave.

2. V skripti vaja09.py najdete deklaracijo funkcije za afino poravnavo kontrolnih točk:

```
def affineRegistration(iType, rCP, iCP, iImage):
    """
    Funkcije za afino poravnavo kontrolnih tock
    """

    raise NotImplementedError("Implement me")

    return oT, oCP, oImage
```

kjer vhodni argument `iType` predstavlja vrsto afine poravnave (`'interpolation'` za interpolacijsko, `'approximation'` za aproksimacijsko), `rCP` $= [x_1, y_1;\ x_2, y_2;\ \ldots;\ x_K, y_K]$ točke na referenčni sliki in `iCP` $= [u_1, v_1;\ u_2, v_2;\ \ldots;\ u_K, v_K]$ pripadajoče točke na vhodni sliki, `iImage` $= b(u, v)$ pa vhodno sliko. Izhodni argument `oT` predstavlja matriko afine preslikave, `oCP` $= [x'_1, y'_1;\ x'_2, y'_2;\ \ldots;\ x'_K, y'_K]$ poravnane točke iz vhodne slike na referenčno sliko, `oImage` $= b(x, y)$ pa poravnano vhodno sliko.

Privzeto implementacijo dane funkcije nadomestite z delujočo implementacijo tako, da bo funkcija dejansko izvedla afino interpolacijsko poravnavo:

$$\mathbf{T} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix}^{-1}$$

oziroma afino aproksimacijsko poravnavo:

$$\overline{x} = \frac{1}{K}\sum_{k=1}^{K} x_k; \quad \overline{xx} = \frac{1}{K}\sum_{k=1}^{K} x_k x_k; \quad \overline{ux} = \frac{1}{K}\sum_{k=1}^{K} u_k x_k; \quad \ldots$$

$$\begin{bmatrix} \overline{xx} & \overline{xy} & \overline{x} & 0 & 0 & 0 \\ \overline{xy} & \overline{yy} & \overline{y} & 0 & 0 & 0 \\ \overline{x} & \overline{y} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \overline{xx} & \overline{xy} & \overline{x} \\ 0 & 0 & 0 & \overline{xy} & \overline{yy} & \overline{y} \\ 0 & 0 & 0 & \overline{x} & \overline{y} & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ t_x \\ a_{21} \\ a_{22} \\ t_y \end{bmatrix} = \begin{bmatrix} \overline{ux} \\ \overline{uy} \\ \overline{u} \\ \overline{vx} \\ \overline{vy} \\ \overline{v} \end{bmatrix}$$

$$\mathbf{P}_{xy}\,\mathbf{t} = \mathbf{p}_{uv}$$

$$\mathbf{t} = \mathbf{P}_{xy}^{-1}\,\mathbf{p}_{uv} \quad \Rightarrow \quad \mathbf{T}$$

Za preslikavo slike uporabite funkcijo `transformImage()` [Vaja 6: Geometrijske preslikave slik], pri čemer za ozadje slike izberete sivinsko vrednost $s = 0$, za določanje ostalih sivinskih vrednosti pa interpolacijo prvega reda.

3. V skripti vaja09.py najdete deklaracijo funkcije za izračun rezultatov poravnave v obliki slike razlik in slike šahovnice:

```
def getImages(iImage1, iImage2, iStep):
    """
    Funkcije za izracun rezultatov poravnave v obliki slike razlik
   in slike sahovnice
    """

    raise NotImplementedError("Implement me")

    return oImage1, oImage2
```

kjer vhodni argument `iImage1` predstavlja prvo (referenčno) sliko, `iImage2` drugo (poravnano vhodno sliko), `iStep` pa velikost polja šahovnice. Izhodni argument `oImage1` predstavlja sliko razlik med prvo in drugo sliko, `oImage2` pa sliko šahovnice prve in druge slike.

Privzeto implementacijo dane funkcije nadomestite z delujočo implementacijo tako, da bo funkcija dejansko izračunala sliko razlik in sliko šahovnice.

Gradivo

Naslednje naloge so neobvezne in namenjene boljšemu razumevanju vsebine.

1. V skripti vaja09.py najdete deklaracijo funkcije za izračun napake poravnave:

```
def computeError(rCP, iCP, oCP, rImage, iImage, oImage, iArea):
    """
    Funkcije za izracun napake poravnave
    """

    raise NotImplementedError("Implement me")

    return R2, MSE
```

kjer vhodni argument `rCP` $= [x_1, y_1;\ x_2, y_2;\ \ldots;\ x_K, y_K]$ predstavlja točke na referenčni sliki, `iCP` $= [u_1, v_1;\ u_2, v_2;\ \ldots;\ u_K, v_K]$ pripadajoče točke na vhodni sliki, `oCP` $= [x'_1, y'_1;\ x'_2, y'_2;\ \ldots;\ x'_K, y'_K]$ poravnane točke iz vhodne slike na referenčno sliko, `rImage` $= a(x, y)$ referenčno sliko, `iImage` $= b(u, v)$ vhodno sliko, `oImage` $= b(x, y)$ poravnano vhodno sliko, `iArea` $= [x, y, w, h]$ pa pravokotno območje v koordinatnem sistem referenčne slike z zgornjim desnim ogliščem na slikovnem elementu (x, y) ter širine w in višine h slikovnih elementov (vrednosti lahko nastavimo v uporabniškem vmesniku). Izhodni argument `R2` $= [R_1^2, R_2^2]$ predstavlja povprečno kvadratno razdaljo med vsemi pari kontrolnih točk pred (R_1^2) in po (R_2^2) poravnavi, `MSE` $= [MSE_1, MSE_2]$ pa srednjo kvadratno napako sivinskih vrednosti slike na območju `iArea` pred (MSE_1) in po (MSE_2) poravnavi. Privzeto implementacijo dane funkcije nadomestite z delujočo implementacijo tako, da bo funkcija dejansko izračunala povprečno kvadratno razdaljo kontrolnih točk R^2 in srednjo kvadratno napako sivinskih vrednosti MSE:

$$R^2 = \frac{1}{K} \sum_{k=1}^{K} \Big(\mathcal{T}(x_k, y_k) - (u_k, v_k) \Big)^2 .$$

$$MSE = \frac{1}{IJ} \sum_{i=1}^{I} \sum_{j=1}^{J} \Big(a(x_i, y_j) - b(x_i, y_j) \Big)^2 ,$$

kjer je I število stolpcev in J število vrstic izbrane slike, ozadje slik pa je določeno z $s = 0$. Pri izračunu MSE_2 upoštevajte slikovne elemente, ki niso del ozadja (torej ne upoštevajte slikovnih elementov s sivinsko vrednostjo $s = 0$ na poravnani sliki ter njihovih pripadajočih parov na referenčni sliki). Izračunane vrednosti, ki naj bodo zaokrožene na dve decimalni mesti, bo uporabniški vmesnik ustrezno izpisal.

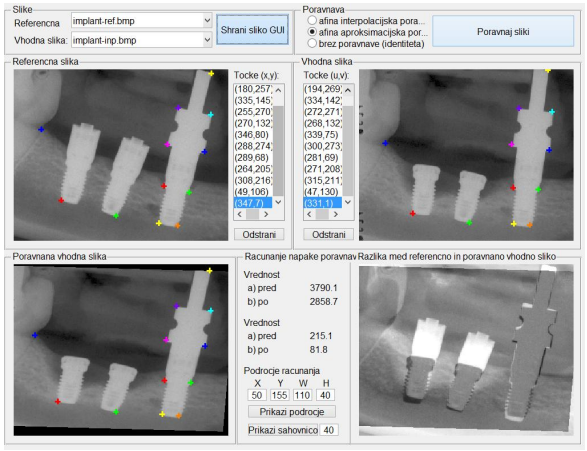
2. Na podlagi ročnega določanja točk na referenčni in vhodni sliki preko uporabniškega vmesnika izvedite naslednje poravnave:

   (a) določite $K = 3$ pare kontrolnih točk in izvedite afino interpolacijsko poravnavo;

   (b) določite $K = 6$ parov kontrolnih točk in izvedite afino aproksimacijsko poravnavo;

   (c) določite $K = 12$ parov kontrolnih točk in izvedite afino aproksimacijsko poravnavo.

   Za vsako od zgoraj navedenih poravnav izpišite pripadajoče slike uporabniškega vmesnika (s prikazanimi točkami, poravnano sliko, sliko razlik ter izpisanimi vrednostmi R^2 in MSE), ki jih pridobite s pritiskom na gumb "Shrani sliko GUI".



Slika 1: afina aproksimacijska preslikava s K = 12 parov kontrolnih točk

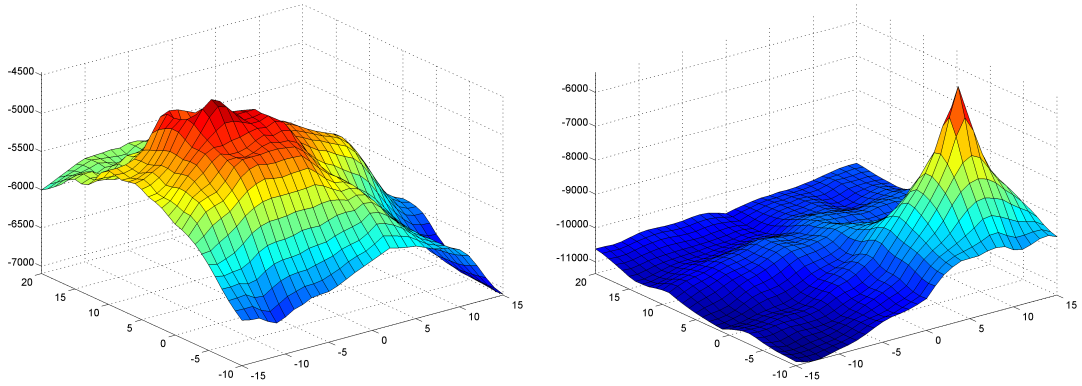
3. Dana je referenčna slika head-T2-083x100-08bit.raw ter dve testni vhodni sliki head-T1-083x100-08bit.raw in head-SD-083x100-08bit.raw velikosti $X \times Y = 83 \times 100$ slikovnih elementov ter velikostjo slikovnega elementa $\Delta X \times \Delta Y = 1\,\text{mm} \times 1\,\text{mm}$, ki so zapisane v obliki surovih podatkov (RAW) z 8 biti na slikovni element.

   Napišite funkcijo za geometrijsko poravnavo z optimizacijo podobnosti na podlagi izčrpnega iskanja največje podobnosti med referenčno in izbrano vhodno sliko glede na parametre premika t_x in t_y:

```
def exhaustiveRegistration(iImageA, iImageB, iTx, iTy):
    # ...
    return oMap, oTx, oTy
```

   kjer vhodni argument `iImageA` $= a(x, y)$ predstavlja referenčno sliko, `iImageB` $= b(u, v)$ vhodno sliko, `iTx` $= [t_{min}, t_{max}, \Delta t]$ oz. `iTy` $= [t_{min}, t_{max}, \Delta t]$ pa vektorja parametrov premikov v x oz. y smeri (začetni premik t_{min}, končni premik t_{max}, korak premika Δt; vse v milimetrih). Izhodni argument `oMap` predstavlja matriko porazdelitve izračunane mere podobnosti MP v prostoru premikov, tako da ima element (i, j) te matrike vrednost $\text{MP}\big(a(x, y), b(x, y)\big)|_{t_x(i), t_y(j)}$, `oTx` oz. `oTy` pa vektorja vseh izvedenih premikov v x oz. y smeri. Za preslikavo uporabite funkcijo `transformImage()` [Vaja 6: Geometrijske preslikave slik], za mero podobnosti MP pa uporabite srednjo kvadratno napako MSE.

   izrišite porazdelitve izračunane mere podobnosti v prostoru premikov za obe vhodni sliki na območju premika $t_x = -15 \ldots +15\,\text{mm}$ po koraku $2\,\text{mm}$ in $t_y = -10 \ldots +20\,\text{mm}$ po koraku $3\,\text{mm}$ ter zapišite optimalne parametre poravnave.



Slika 2: Porazdelitve mere podobnosti v prostoru premikov za obe sliki