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

NAVODILA

Vaja 4: Prikazovanje 3D slik v 2D

Pripravila: Gašper Podobnik & Tomaž Vrtovec

Navodila

Objekti, ki jih opazujemo v dvodimenzionalnih (2D) slikah, so v splošnem tridimenzionalni (3D) in jih lahko zato boljše analiziramo na podlagi 3D slik, ki jih predstavimo kot funkcije $f(x, y, z)$ slikovnih vrednosti (intenzitet) v odvisnosti od prostorskih koordinat x, y in z. Prikazovanje 3D slik v 2D omogočajo načini prikazovanja v obliki *prerezov* in *projekcij*.

1. Dana je 3D sivinska slika spine-512x058x907-08bit.raw velikosti $X \times Y \times Z = 512 \times 58 \times 907$ slikovnih elementov, ki je zapisana v obliki surovih podatkov (RAW) z 8 biti na slikovni element, velikost slikovnega elementa pa je $\Delta x \times \Delta y \times \Delta z = 0{,}597656 \times 3 \times 0{,}597656\,\text{mm}$. Slika je bila zajeta s tehniko računalniške tomografije (CT) in prikazuje človeško telo.

   Napišite funkcijo za nalaganje 3D slike:

   ```
   def loadImage3D(iPath, iSize, iType):
       # ...
       # your code goes here
       # ...
       return oImage
   ```

   kjer vhodni argument `iPath` predstavlja pot do datoteke s sliko (mapa in ime datoteke), `iSize` vektor velikosti slike (v slikovnih elementih), `iType` pa obliko zapisa (vrsta podatkov). Izhodni argument `oImage` predstavlja 3D matriko slikovnih elementov.

   Naložite dano sliko.

2. Na osnovi 3D slike $f(x, y, z)$ lahko določimo pravokotne ravninske prereze, in sicer:

   - stranski (lateralni, sagitalni): $f_{x=x_c}(y, z) = f(x_c, y, z)$,
   - čelni (frontalni, koronalni): $f_{y=y_c}(x, z) = f(x, y_c, z)$,
   - prečni (transverzalni, aksialni): $f_{z=z_c}(x, y) = f(x, y, z_c)$,

   kjer so x_c, y_c in z_c izbrani položaji posameznih prerezov.

   Napišite funkcijo za določanje poljubnega pravokotnega ravninskega prereza:

   ```
   def getPlanarCrossSection(iImage, iDim, iNormVec, iLoc):
       # ...
       # your code goes here
       # ...
       return oCS, oH, oV
   ```

   kjer vhodni argument `iImage` predstavlja 3D sliko, `iDim` velikost slikovnih elementov, `iNormVec` normalni vektor ravnine prereza ($\vec{n}_x = (1, 0, 0)$ za stransko, $\vec{n}_y = (0, 1, 0)$ za čelno ali $\vec{n}_z = (0, 0, 1)$ za prečno ravnino), `iLoc` pa položaj prereza v 3D sliki (x_c, y_c oz. z_c, v slikovnih elementih). Izhodni argument `oCS` predstavlja želeni pravokotni ravninski prerez, `oH` in `oV` pa vektorja položajev slikovnih elementov v vodoravni oz. navpični smeri prereza.

   Prikažite stranski pravokotni ravninski prerez dane 3D slike pri $x_c = 290$.

3. Na osnovi 3D slike $f(x,y,z)$ lahko določimo pravokotne ravninske projekcije, in sicer:
   - stranska (lateralna, sagitalna): $p_x(y,z) = \underset{k}{PF}\big(f(k,y,z)\big); \quad k = 1,2,\ldots,X,$
   - čelna (frontalna, koronalna): $p_y(x,z) = \underset{k}{PF}\big(f(x,k,z)\big); \quad k = 1,2,\ldots,Y,$
   - prečna (transverzalna, aksialna): $p_z(x,y) = \underset{k}{PF}\big(f(x,y,k)\big); \quad k = 1,2,\ldots,Z,$

   kjer je PF poljubna funkcija točk. Med najbolj razširjenimi vrstami projekcij je t. i. projekcija maksimalnih vrednosti oz. MIP (*ang.* maximal intenisty projection), ki za funkcijo točk uporablja maksimalno vrednost.

   Napišite funkcijo za določanje poljubne pravokotne ravninske projekcije:

```
def getPlanarProjection(iImage, iDim, iNormVec, iFunc):
    # ...
    # your code goes here
    # ...
    return oP, oH, oV
```

   kjer vhodni argument `iImage` predstavlja 3D sliko, `iDim` velikost slikovnih elementov, `iNormVec` normalni vektor ravnine projekcije ($\vec{n}_x$, $\vec{n}_y$ ali $\vec{n}_z$), `iFunc` pa funkcijo točk PF v obliki funkcije iz knjižnice `numpy` (npr. `np.max` za MIP). Izhodni argument `oP` predstavlja želeno pravokotno ravninsko projekcijo, `oH` in `oV` pa vektorja položajev slikovnih elementov v vodoravni oz. navpični smeri prereza.

   Prikažite stransko pravokotno ravninsko projekcijo vrste MIP.

Vprašanja

Odgovore na sledeča vprašanja zapišite v poročilo, v katerega vstavite zahtevane izrise in programske kode.

1. Priložite slike naslednjih pravokotnih ravninskih prerezov dane 3D slike:
   - stranski prerez na položaju $x_c = 256$ slikovnih elementov,
   - čelni prerez na položaju $y_c = 35$ slikovnih elementov,
   - prečni prerez na položaju $z_c = 467$ slikovnih elementov.

   Priložene slike naj bodo v dimenzijskem sorazmerju z velikostjo slikovnega elementa dane 3D slike.

2. Priložite slike stranske, čelne in prečne pravokotne ravninske projekcije dane 3D slike, pri čemer za funkcijo točk PF uporabite:
   - maksimalno vrednost,
   - povprečno vrednost.

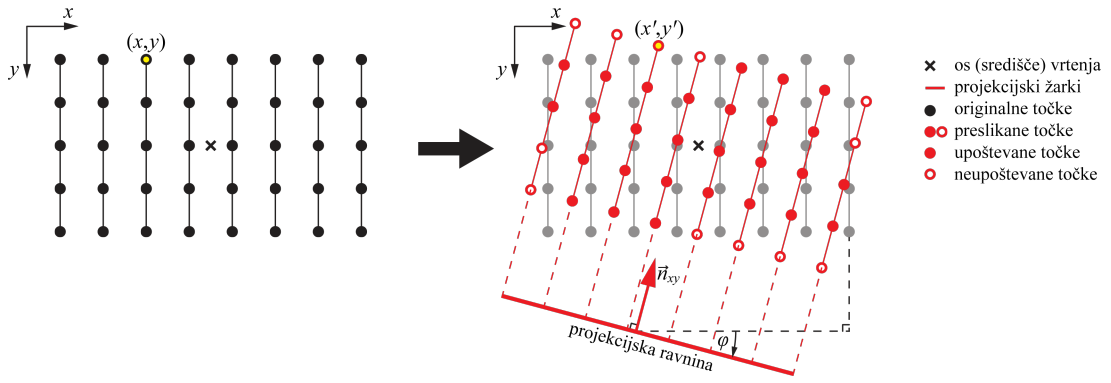
   Priložene slike naj bodo v dimenzijskem sorazmerju z velikostjo slikovnega elementa dane 3D slike.

3. Katere vrste projekcij, pri katerih za funkcijo točk uporabite maksimalno vrednost, minimalno vrednost, povprečno vrednost, vrednost mediane, vrednost standardnega odklona oz. vrednost variance, je v primeru prikazovanja CT slik človeškega telesa sploh smiselno računati? Obrazložite odgovor.

4. Določite poljubno čelno poševno ravninsko projekcijo na podlagi normalnega vektorja na ravnino projekcije v obliki $\vec{n}_{xy} = (n_x, n_y, 0)$, ki določa kot rotacije φ ravnine projekcije. Uporabite najbolj preprost pristop k določevanju poševne projekcije, in sicer da koordinate točk oz. slikovnih elementov (x, y) v čelnem pravokotnem pogledu povežete v daljice ter izračunate koordinate zavrtenih točk (x', y'):

$$x' = x\cos\varphi - y\sin\varphi,$$
$$y' = x\sin\varphi + y\cos\varphi,$$

pri čemer os (središče) vrtenja postavite v središče prečne ravnine. Projekcije nato računate vzdolž posameznih zavrtenih daljic, pri čemer ne upoštevate točk, ki so izven slike. Za interpolacijo sivinskih vrednosti lahko uporabite princip najbližjega soseda, velikost slikovnega elementa pa lahko ohranite enako kot v primeru čelne pravokotne ravninske projekcije. Glede na to, da gre za čelno projekcijo, postopek ponovite za vsak prerez vzdolž koordinatne osi z.



Algoritem implementirate v obstoječi funkciji `getPlanarProjection()` tako, da dodate naslednji pogojni stavek:

```
elif iNormVec(3) == 0:
    # ...
    # your code goes here
    # ...
```

Priložite programsko kodo algoritma ter slike čelnih poševnih ravninskih projekcij vrste MIP za normalne vektorje $\vec{n}_1 = (3{,}83,\ 9{,}24,\ 0)$, $\vec{n}_2 = (1,\ 1,\ 0)$ in $\vec{n}_3 = (9{,}24,\ 3{,}83,\ 0)$, ki so v dimenzijskem sorazmerju z velikostjo slikovnega elementa dane 3D slike. Naštejte in obrazložite nekaj pomanjkljivosti uporabljenega pristopa k določanju poševnih ravninskih projekcij.

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati k poročilu, prispevajo pa naj k boljšemu razumevanju vsebine.

Upodabljanje površine združuje postopke prikazovanja površine danega 3D objekta. Ena izmed najbolj razširjenih metod za upodabljanje površine je *triangulacija površine*, v Matlabu pa je implementirana na podlagi t. i. funkcije izo-površine (*ang.* iso-surface), ki medsebojno poveže vse točke v prostoru z enako izo-vrednostjo (tj. sivinska vrednost, ki določa mejo med objektom in ozadjem v sliki). S pomočjo spodnje funkcije za triangulacijo površine `surfaceTriangulation()`, ki temelji na računanju izo-površine, poskusite upodobiti površino v 3D sliki `iImage` z velikostjo slikovnega elementa `iDim` pri različnih izo-vrednostih `iIsoValue`:

```
from skimage import measure
from mpl_toolkits.mplot3d import Axes3D, art3d
from matplotlib.colors import LightSource


def surfaceTriangulation(iImage, iDim, iIsoValue, step_size: int = 3):
    """Funkcija za triangulacijo povrsine in graficni izris
    Args:
        iImage (np.ndarray): vhodna slika
        iDim (list, tuple): velikost slikovnega elementa, (dx, dy, dz)
        iIsoValue (int): izo vrednost
        step_size (int): Mminimalno 1 za najbolj natancen izris, ampak
   racunsko potratno, mora biti tipa int. Defaults to 3.
    """
    # zapri vsa okna
    plt.close("all")

    # spacing (dy, dx, dz)
    spacing = np.array([iDim[1], iDim[0], iDim[2]])
    # izracun in izris izo-povrsine
    verts, faces, _, _ = measure.marching_cubes(
        volume=iImage, level=iIsoValue, step_size=step_size, spacing=
   spacing
    )

    view_altdeg = 20
    # view_azdeg = 217.5
    view_azdeg = 217.5 - 180

    fig = plt.figure()
    ax = fig.add_subplot(111, projection="3d")
    title = f"Triangulacija povrsine za izo-vrednost: {iIsoValue}"
    ax.set_title(title)
    ax.set_xlim(verts[:, 0].max(), 0)
    ax.set_ylim(verts[:, 1].max(), 0)
    ax.invert_zaxis()
    ls = LightSource(altdeg=view_altdeg, azdeg=view_azdeg)
    ax.plot_trisurf(
        verts[:, 0], verts[:, 1], faces, verts[:, 2], color="red",
   lightsource=ls
    )
    ax.view_init(elev=view_altdeg, azim=view_azdeg)
    plt.show()
```


POMISLI,
PREDEN
NATISNEŠ