PRILOGE

UPORABLJEN PROGRAM

```
# -*- coding: utf-8 -*-

import os
import numpy as np
import pylab as pl
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter


#spremeni working dir (datoteke shrani v enako mapo kot se nahaja ta datoteka)
full_path = os.path.realpath(__file__)
this_dir = str(os.path.dirname(full_path))
os.chdir(this_dir)

# kart_nor = open("Osnovno_oct10mm_nekaj.txt","r")
# k_nor = kart_nor.readlines()

def sfr_max (st_raz, k_nor, izris = 0):
    '''
    st_raz(int) = stevilo razredov(vecje - bolj natanca segmentacija)
    st_objektov = int, kolkim objektom naj izvozi indekse iz oblaka tock
    k_nor = list stringov, prebran oblak tock npr. ["x y z xn yn zn", " "," "]

    Vrne matriko s stevilom normal v celici ter matriko z indeksi normal iz
    izvornega oblaka tock v vsaki celici
    '''
    xn = []
    yn = []
    zn = []
    fis = []
    lams = []
    k  = 0


    #postavitev spremenljivk za shranjevanje
    raz = []
    coun = []
    for i in range(st_raz):
        raz.append([])
        coun.append([])
        for j in range(st_raz):
            raz[i].append([])
            coun[i].append(0)

    #prebere kartezicne koordinate normal iz oblaka tock
    # in jih shrani po oseh x, y, z

    for i in range(len(k_nor)):
        k +=1
        vrstica = k_nor[i]
        vrstica = vrstica.split()
        xi = float(vrstica[3])
        yi = float(vrstica[4])
        zi = float(vrstica[5])


        xn.append (xi)
        yn.append (yi)
        zn.append (zi)
    print (k)

    #za vsako točko iz normal izračuna sferne koordinate normale na gausovi
    #sferi in jih porazdeli v st_raz(int)^2 celic. Za vsak razred shrani indeks
    #tock iz te celice v raz spremenljivko. coun deluje kot stevec tock v vsaki
```

celici

```
    fik = 90/st_raz
    lamk = 360/st_raz

    for i in range(len(xn)):
        xi = xn[i]
        yi = yn[i]
        zi = zn[i]

        if xi == 0:

            continue

        di = np.sqrt(xi**2 + yi**2)
        R = np.sqrt(xi**2 + yi**2 + zi**2)
        fi_r = np.arcsin(zi/R)
        lam_r = np.arctan2(yi,xi)

        fi = fi_r * 180 / np.pi
        lam = lam_r * 180 / np.pi
        lam = lam + 180


        lams.append(lam)
        fis.append(fi)

        lamp = 0

        for j in range(st_raz):
            if lamp < lam <= lamp+lamk:
                fip = 0
                for k in range(st_raz):
                    if fip < fi <= fip+fik:
                        coun[j][k] +=1
                        raz[j][k].append(i)
                    fip += fik
            lamp += lamk


    # Izris
    if izris != 0:

        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')
        x1 = np.arange(0,360, lamk)
        y1 = np.arange(0,90, fik)

        y1, x1 = np.meshgrid(y1, x1)

        SUR = ax.plot_surface(x1, y1, coun,rstride=1, cstride=1, cmap=cm.coolwarm,
                              linewidth=0, antialiased=False)

        ax.set_xlabel(r'$\lambda$', fontsize=20)
        ax.set_ylabel(r'$\varphi$', fontsize=20)
        ax.set_zlabel('Število normalnih vektorjev', fontsize=15)
        fig.colorbar(SUR, shrink=0.5, aspect=5)
        plt.show()



    return (coun, raz)
```

```
# -*- coding: utf-8 -*-

import os
import numpy as np
import pylab as pl
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from maximum_sfera import sfr_max


#spremeni working dir (datoteke shrani v enako mapo kot se nahaja ta datoteka)
full_path = os.path.realpath(__file__)
this_dir = str(os.path.dirname(full_path))
os.chdir(this_dir)


def binar(rast, edge):
    '''
    rast = matrika s stevilom normal v celici. Oblika je list v list [[],[],[],[]]
    edge = celo stevilo. kakšna je meja za spremembo binarne vrednosti

    Vzame vhodno matriko in glede na mejo določi binarne vrednosti
    Vrne matriko enakih dimenzij z binarnimi vrednostmi glede na mejo
    '''

    val = rast.copy()

    # Glasovanje
    for i in range(len(val)):
        for j in range(len(val[0])):
            k = val[i][j]
            if k <= edge:
                val[i][j] = 0
            else:
                val[i][j] = 1


    # Izris
    val1 = np.asarray(val)

    fig= plt.figure()
    ax1 = plt.axes()
    y1 = np.linspace(0,360, len(val))
    x1 = np.linspace(0,90, len(val[0]))

    x1, y1 = np.meshgrid(x1, y1)

    plt.imshow(val1, cmap ='Greys_r' )

    plt.show()

    return val

def hough_line(img):
  '''
  img = binarna matrika oblike [[],[],[],[]]

  Izracuna indekse parametrov linije in vrne v kombinaciji s stevilom glasov
  za posamezen par parametrov. Vrne tudi vektorja dolzin in kotov.
  '''

  # vektorja dolžin in kotov
  thetas = np.deg2rad(np.linspace(-90.0, 90.0, 2*90))
  dim = len(img)
  diag_len = int(np.ceil(np.sqrt(dim*dim + dim*dim))  ) # maksimalna dolžina
```

```
    rhos = np.linspace(-diag_len, diag_len, diag_len * 2.0)
    # Vrednosti za nadaljno uporabo
    cos_t = np.cos(thetas)
    sin_t = np.sin(thetas)
    num_thetas = len(thetas)

    # Matrika parametricnega prostora
    accumulator = np.zeros((num_thetas, 2*diag_len))
    accumulator = accumulator.tolist()

    # Indeksi mejnih tock (celic), skozi katere zelimo iskati linijo

    y_idxs, x_idxs = np.nonzero(img)


    # Glas v matriki parametricnega prostora
    for i in range(len(x_idxs)):
      x = x_idxs[i]
      y = y_idxs[i]

      for t_idx in range(num_thetas):
        # Izracun rho. Maksimalna dolzina dodana za pozitiven indeks
        rho = int(round(x * cos_t[t_idx] + y * sin_t[t_idx]) + diag_len)
        accumulator[t_idx][rho] += 1


    #sortiranje
    vred = []
    ind = []

    for j in range(len(accumulator)):
        for k in range(len(accumulator[0])):
            vred.append(accumulator[j][k])
            ind.append([j,k])


    max_ind = sorted (zip(vred, ind), reverse = True)

    return max_ind, rhos, thetas


def local_max (max_ind, rhos, thetas, ods):
    '''
    max_ind, rhos, thetas so izhodni podatki funkcije
    odst = float ali int

    iskanje lokalnih maximumov glede na odstopanje po dolzini
    '''

    locals = []
    locals.append(max_ind[0])

    for i in range(len(max_ind)):
        if locals[-1][1][1] - max_ind[i][1][1] >= ods:
            locals.append(max_ind[i])

    return locals


def get_cells (locals, rast, loc_num, thetas, rhos):
    '''
    locals = izhodni podatek funkcije local_max
    rast = izhodni podatek funkcije sfr_max
    loc_num = int, številka željenega lokalnega maksimuma po vrsti
    thetas, rhos = vektorja dolžin in kotov, izhodni podatek hough_line
```

```
Vrne indekse celic matrike oblike rast skozi katere poteka linija v dveh
enako velikih vektorjih.
'''
dim = len(rast)
diag = int(np.ceil(np.sqrt(dim*dim + dim*dim))  )

#iskanje pravih parametrov
rho_ind = locals[loc_num][1][1]
theta_ind = locals[loc_num][1][0]

rho = rhos[rho_ind]
theta = np.deg2rad(thetas[theta_ind])


#generiranje vektorja za izračun
ys = (np.linspace(0, len(rast)-1, len(rast))).tolist()
xs = []
# izracun y-ov
for yi in ys:
    x = ( (rho-yi*np.sin(theta))/(np.cos(theta)) )
    xs.append(int(round(x)))

return xs, ys
```

```
# -*- coding: utf-8 -*-

import os
import numpy as np
import pylab as pl
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


#spremeni working dir (datoteke shrani v enako mapo kot se nahaja ta datoteka)
full_path = os.path.realpath(__file__)
this_dir = str(os.path.dirname(full_path))
os.chdir(this_dir)


def v_list(obj_ot):
    '''
    Spremeni readlines metodo branja v 3 liste s koordinatami x, y in z
    '''
    obj_x = []
    obj_y = []
    obj_z = []

    for i in range(len(obj_ot)):
        temp1 = obj_ot[i].split()[0:3]
        obj_x.append(float(temp1[0]))
        obj_y.append(float(temp1[1]))
        obj_z.append(float(temp1[2]))

    return [obj_x, obj_y, obj_z]


def search_plot1 ( OT, bins, x_ras, y_ras):
    '''
    maxs, list z tupli, ki vsebujejo število točk v enem izmed max razredov
    in indexe teh točk kot drugi element v tuplu (st_tock, [indeks])

    ind je št.objekta, ki ga iščem

    v oblaku točk poišče točke z indeksi iz max razredov

    vrne koordinate točk iz oblaka tock
    '''

    indeksi_tock0 = []
    lenx = len(x_ras)

    # Vzame vse indekse tock in jih pospravi v vektor(list)
    for i in range(lenx):
        indeksi_tock0.append( bins [int(y_ras[i])-1] [int(x_ras[i])-1] )

    indeksi_tock = sum(indeksi_tock0, [])

    # Sestavi oblak tock iz orginalnega preko danih indeksov
    obj_ot = []
    for i in indeksi_tock:
        obj_ot.append(OT[i])

    # Spremeni iz string v float matriko
    matrika = v_list(obj_ot)

    #izris tock iz posameznega razreda
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    plt.axis('equal')
```

```
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
ax.scatter(matrika[0], matrika[1], matrika[2])

ax.get_xaxis().set_ticks([])
ax.get_yaxis().set_ticks([])
# ax.get_zaxis().set_ticks([])

ax.set_axis_off()

plt.show()

return matrika
```

```
# -*- coding: utf-8 -*-

import os
import numpy as np
import pylab as pl
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from maximum_sfera import sfr_max
from bin_hough import binar, hough_line, local_max, get_cells
from search_plot import search_plot1

#spremeni working dir (datoteke shrani v enako mapo kot se nahaja ta datoteka)
full_path = os.path.realpath(__file__)
this_dir = str(os.path.dirname(full_path))
os.chdir(this_dir)

#Odpri in preberi vhodno datoteko [x,y,z,xn,yn,zn]
kart_nor = open("Osnovno_oct10mm_nekaj.txt","r")
k_nor = kart_nor.readlines()

#Izračunaj matriko z zgoščišči na sferi in shrani rezultate v ločeni
#spremenljivki
podat = sfr_max(45,k_nor,1)
rast = podat[0]
bins_s_tock = podat[1]

#Izdelaj binarno matriko
k = binar(rast, 10)

#Izraunaj parametre linij v lokalnem maksimumu
max_ind, rhos, thetas= hough_line(k)
locals = local_max(max_ind, rhos, thetas, 15)

#Pridobi celice, ki jih linija seka
x_rav, y_rav = get_cells(locals, rast, 0, thetas, rhos)
x_stoz, y_stoz = get_cells(locals, rast, 1, thetas, rhos)
x_valj, y_valj = get_cells(locals, rast, 2, thetas, rhos)

#S pomočjo teh celic pridobi indekse iz orginalnega oblaka in izriši
plot1 = search_plot1(k_nor, bins_s_tock, x_stoz, y_stoz)
plot2 = search_plot1(k_nor, bins_s_tock, x_valj, y_valj)
plot3 = search_plot1(k_nor, bins_s_tock, x_rav, y_rav)
```