[1]
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy import ndimage
import os
import SimpleITK as sitk
```

Defining a user data type: each pixel in the axial view with respect to the beam is associated to an axial_spot(x,y).

Field

'flag': dtype = boolean True if the that pixel belongs to the DDS mask and the integrated activity along that profile is over than 20% of maximum integrated activity in both the considered images. Otherwise, it's set to False. 'z_max_PET': dtype = int32 Coordinate in voxel dimension where the normalized activity intensity is 0.2 in the PET image along beam direction. 'z_prox': dtype = int32 Coordinate in voxel dimension where the nomralized activity reaches its maximum (i.e., 1.0) along beam direction. 'z_dist': dtype = int32 It's defined as z_max - 5mm 'A1_interp', 'A2_interp': dtype = Python-Object Interpolate 1D spline of the considered profiles in the first and second image compared. 'delta_MLS': dtype = 'float' Range difference computed as the most-likely shift. 'delta_stb': dtype = 'float' Parameter related to the stability of the minimum and to the reliability of the found range difference.

[2]
```
axial_spot = np.dtype([('flag', '?'),
                       ('z_max_PET', 'i4'),
                       ('z_prox', 'i4'),
                       ('z_dist','i4'),
                       ('A1_interp', 'O'),
                       ('A2_interp', 'O'),
                       ('delta_MLS', 'f4'),
                       ('delta_stab', 'f4')
                      ])
```

PATIENT MAIN FOLDER

[3]
```
patient_folder = '/Users/gretadelnista/Desktop/INSIDE/Data/Trial/006P/'
#SCC = '/Users/gretadelnista/Desktop/INSIDE/Data/Simulazione_SCC_changed'
```

PET image folder

[4]
```
PET_folder = os.path.join(patient_folder, 'PET_measurements/')
compared = '2vs21'
```

Data loading and reshaping: PET 1

```
[5]  PET_1 = os.path.join(PET_folder, 'PETfraction002/002_iter5subset1.gipl.gz
     #PET_1 = os.path.join(SCC, 'B3_interspill_01.nii')
     PET_1 = sitk.ReadImage(PET_1)
     data_PET1 = sitk.GetArrayFromImage(PET_1)
```

Data loading: PET 2

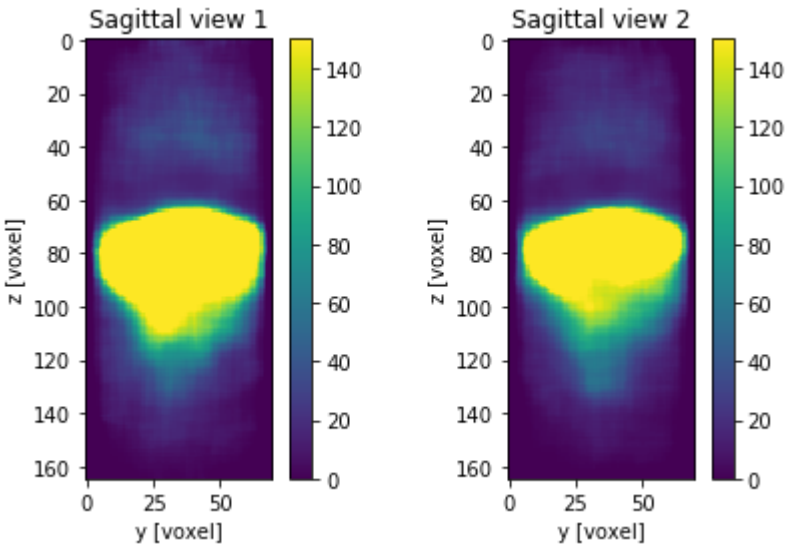
```
[6]  PET_2 = os.path.join(PET_folder, 'PETfraction021/021_iter5subset1.gipl.gz
     #PET_2 = os.path.join(SCC, 'B3_interspill_02.nii')
     PET_2 = sitk.ReadImage(PET_2)
     data_PET2 = sitk.GetArrayFromImage(PET_2)
```

```
[7]  shape_ = PET_1.GetSize()
     irradiated_mask = np.empty(shape = (shape_[1], shape_[0]), dtype = axial_
```

data_PET2 = os.path.join(PET_folder, 'fraction016/interspill_194_297_calibration_iter5subset1.raw') data_PET2 = np.fromfile(data_PET2, dtype=np.float32) data_PET2 = data_PET2.reshape((165, 70, 140)) data_PET1 = os.path.join(PET_folder, 'fraction017/interspill_398_494_calibration_iter5subset1.raw') data_PET1 = np.fromfile(data_PET1, dtype=np.float32) data_PET1 = data_PET1.reshape((165, 70, 140))

Displaying raw data: sagittal view (parallel to beam direction)

```
[28] plt.figure()
     ax1 = plt.subplot(1, 2, 1)
     ax1.set_xlabel('y [voxel]')
     ax1.set_ylabel('z [voxel]')
     ax1.set_title('Sagittal view 1')
     plt.imshow(data_PET1[:, :, 70], vmax = 150)
     plt.colorbar()

     ax2 = plt.subplot(1, 2, 2)
     ax2.set_xlabel('y [voxel]')
     ax2.set_ylabel('z [voxel]')
     ax2.set_title('Sagittal view 2')
     plt.imshow(data_PET2[:, :, 70], vmax = 150)
     plt.colorbar()
     plt.tight_layout()
```



Applying median filter to remove salt-and-pepper noise

data_PET1 = ndimage.median_filter(data_PET1, footprint=np.ones((5,5, 5))) data_PET2 = ndimage.median_filter(data_PET2, footprint = np.ones((5, 5, 5))) plt.imshow(data_PET1[:, :, 70]) plt.xlabel('y voxel') plt.ylabel('z voxel') plt.title('Sagittal view') plt.colorbar()

[9]
```
median = sitk.MedianImageFilter()
median.SetRadius((5, 5, 5))
median.GetRadius()
PET1_median = median.Execute(PET_1)
PET2_median = median.Execute(PET_2)
```

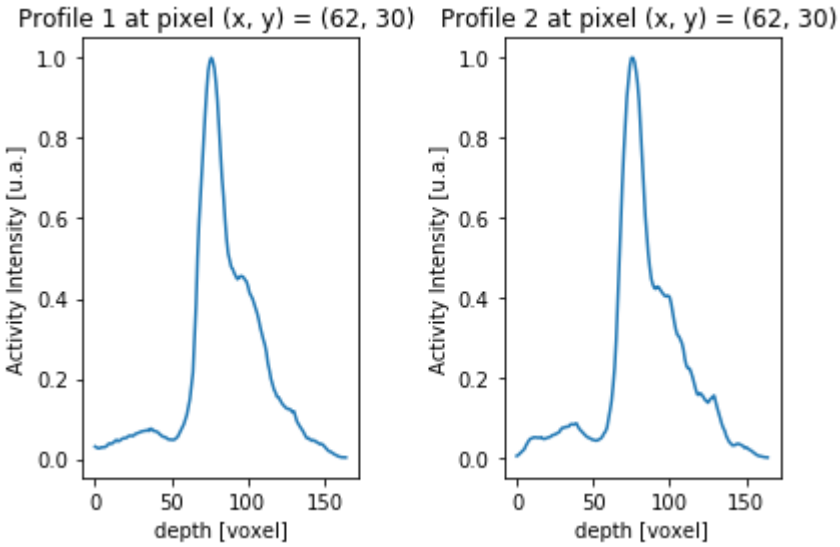
import nibabel as nib data_PET1 = data_PET1.transpose(2,1,0) nib.Nifti1Image(data_PET1, np.eye(4)).to_filename(os.path.join(patient_folder,'fraction016_median.nii'))

[10]
```
data_PET1 = sitk.GetArrayFromImage(PET1_median)
data_PET2 = sitk.GetArrayFromImage(PET2_median)
```

Looking at some profile...

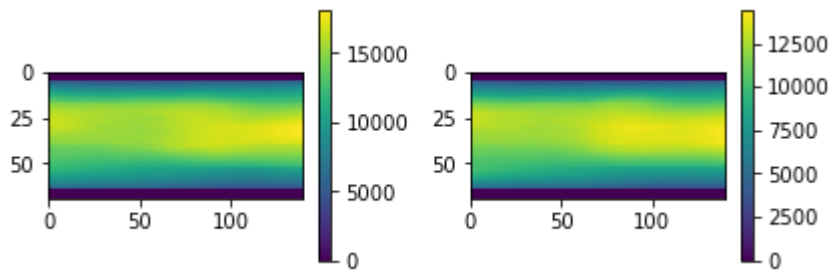
[11]
```
x = 62
y = 30
ax1 = plt.subplot(1,2,1)
plt.plot(data_PET1[:, y, x]/np.amax(data_PET1[:, y, x]))
ax1.set_xlabel('depth [voxel]')
ax1.set_ylabel('Activity Intensity [u.a.]')
ax1.set_title('Profile 1 at pixel (x, y) = ({}, {}) '.format(x, y))

ax2 = plt.subplot(1,2,2)
```

```
plt.plot(data_PET2[:, y, x]/np.amax(data_PET2[:, y, x]))
ax2.set_xlabel('depth [voxel]')
ax2.set_ylabel('Activity Intensity [u.a.]')
ax2.set_title('Profile 2 at pixel (x, y) = ({}, {}) '.format(x, y))

plt.tight_layout()
```



Computing integrated activity along z-axis and plotting where it's higher than 20% of its maximum

[12]
```
P1_sum = np.sum(data_PET1, axis=0)
P2_sum = np.sum(data_PET2, axis=0)
mask = np.logical_and((P1_sum > 0.2*np.max(P1_sum)),  (P2_sum > 0.2*np.ma
plt.figure()
plt.subplot(2, 2, 1)
plt.imshow(P1_sum * mask)
plt.colorbar()

plt.subplot(2, 2, 2)
plt.imshow(P2_sum * mask)
plt.colorbar()
plt.tight_layout()
```



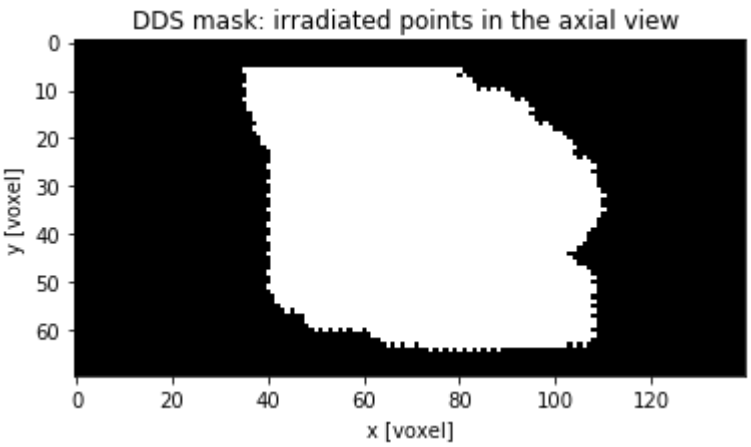
Loading DDS mask

[13]
```
DDS = os.path.join(patient_folder, 'DDS_mask.raw')
#DDS = os.path.join(SCC, 'B3_dds.raw')
DDS = np.fromfile(DDS, dtype = 'bool')
DDS = DDS.reshape((70, 140))
```

Displaying DDS mask

[14]
```
plt.imshow(DDS*mask, cmap='gray')
plt.xlabel('x [voxel]')
plt.ylabel('y [voxel]')
plt.title('DDS mask: irradiated points in the axial view ')
```

```
Text(0.5, 1.0, 'DDS mask: irradiated points in the axial view ')
```



Defining common points between mask and DDS: the 'flag' in irradiated mask is changed.

[15]
```
irradiated_mask['flag'] = np.logical_and(mask, DDS)
```

1) Defining z1: depth where there is 20% fall-off activity (normalized).

The search is restricted at (x,y) present in both DDS_mask and mask definied above (integrated activity along the profile higher than 20% of the maximum ones). The neighborhood of each point identified is controlled: values of the three previous and after points must be higher and lower than 0.2, respectively.

!! I have to decided how to treated the case where len(z1)>1: I think the best choice is to take the first value, since z_max will be definied by the comparision with z_0.5 from CTV.

2) Defining z_prox: depth where there is the maximum activity.

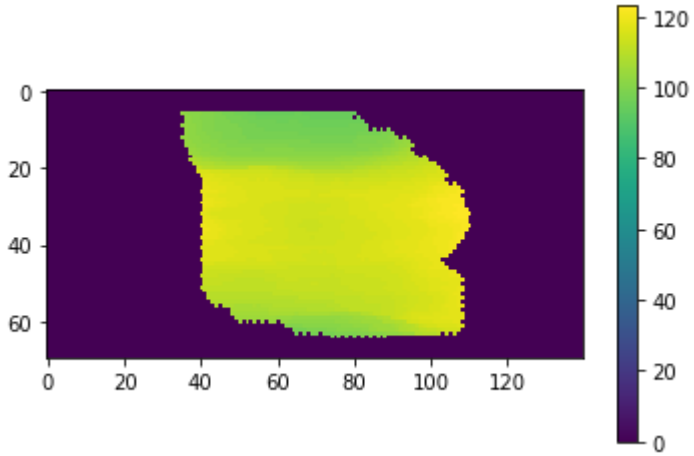
[16]
```
from scipy import interpolate
```

[17]
```
thr = .2
for x_m in range(140):
    for y_m in range(70):
        if irradiated_mask['flag'][y_m, x_m]:
            P1 = data_PET1[:, y_m, x_m]
            A1 = P1 / np.max(P1)
            P2 = data_PET2[:, y_m, x_m]
            A2 = P2 / np.max(P2)
            a = []
            for i in range(2, 163):
                if (A1[i-2:i+1]>=thr).all() and (A1[i+1:i+3]<thr).all():
                    a.append(i)
                    '''
                    if len(a)>1:
                        plt.figure()
                        ax1 = plt.subplot(2, 1, 1)
                        plt.plot(range(165), P1, 'b-,')
                        ax1.set_xlabel('Depth [voxel]')
                        ax1.set_ylabel('Activity intensity [u.a.]')
                        ax2 = plt.subplot(2, 1, 2)
                        plt.plot(range(165), A1, 'r-,')
                        plt.plot(range(165), np.ones(165)*0.2, 'b-')
                        ax1.set_xlabel('Depth [voxel]')
                        ax1.set_ylabel('Normalized Activity intensity [u.
                        plt.suptitle('Profile at pixel (x,y) = ({}, {})'.
                        plt.show()
                    '''
            try:
                irradiated_mask[y_m, x_m]['z_max_PET'] = a[-1]
            except:
                irradiated_mask[y_m, x_m]['z_max_PET'] = np.max([irradiat
                '''
                plt.figure()
                ax1 = plt.subplot(2, 1, 1)
                plt.plot(range(165), A1, 'b-,')
                ax1.set_xlabel('Depth [voxel]')
                ax1.set_ylabel('Normalized Activity intensity 1[u.a.]')
                plt.plot(range(165), np.ones(165)*0.2, 'b-')
                ax2 = plt.subplot(2, 1, 2)
                plt.plot(range(165), A2, 'r-,')
                ax1.set_xlabel('Depth [voxel]')
                ax1.set_ylabel('Normalized Activity intensity 2[u.a.]')
                plt.suptitle('Profile at pixel (x,y) = ({}, {})'.format(x
                plt.show()
```

```
                    '''
                irradiated_mask[y_m, x_m]['z_dist'] = irradiated_mask[y_m, x_
                irradiated_mask[y_m, x_m]['z_prox'] = np.argmax(A1)
                irradiated_mask[y_m, x_m]['A1_interp'] = interpolate.interp1d
                irradiated_mask[y_m, x_m]['A2_interp'] = interpolate.interp1d
```

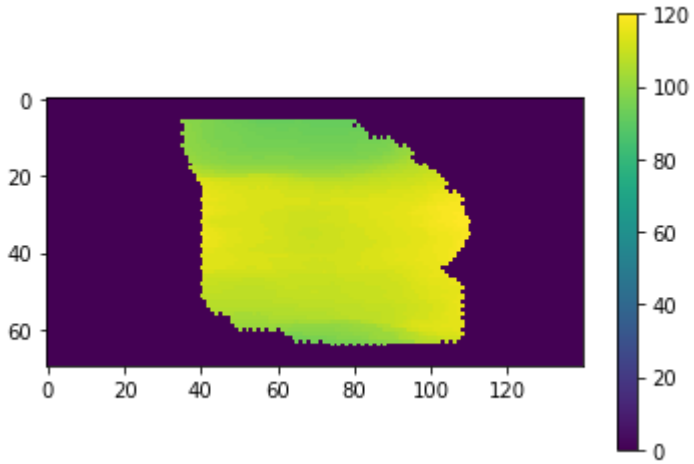
[18]
```
plt.imshow(irradiated_mask['z_max_PET'], vmin = 0)
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x1260070d0>
```



[19]
```
plt.imshow(irradiated_mask['z_dist'], vmin = 0)
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x120e53550>
```



[20]
```
def f_diff(point, zmin, delta):
    I_k = np.arange(zmin, point['z_max_PET'], 1.)
    M_kd = np.tile(I_k, (len(delta), 1)) -  np.reshape(np.repeat(delta, l
    return np.sum(abs(point['A1_interp'](I_k) - point['A2_interp'](M_kd))
```

[21]
```
delta = np.arange(-12, 12, 0.6)
```

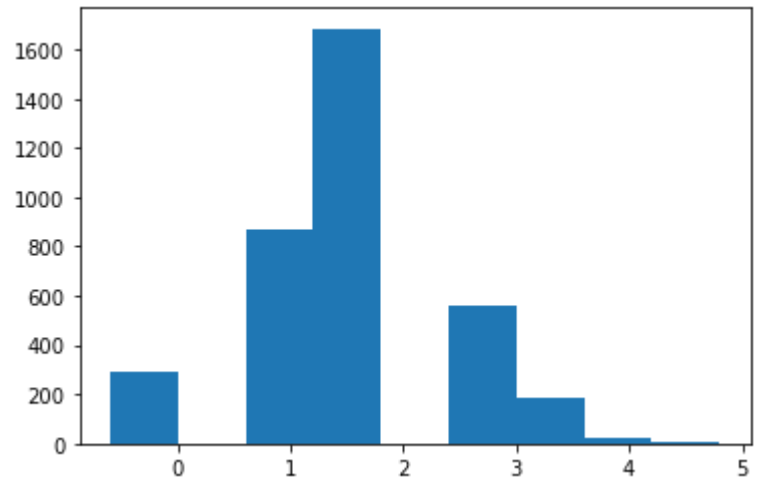
```
for y_m in range(shape_[1]):
    for x_m in range(shape_[0]):
        if irradiated_mask[y_m, x_m]['flag']:
            point = irradiated_mask[y_m, x_m]
            I = np.arange(point['z_prox'], point['z_dist'] + 1, 1.)
            f_diff_delta = np.zeros(shape=(len(I),len(delta)))
            D_diff = np.zeros(len(I))
            for i, z_min in enumerate(I):
                f_diff_delta[i][:] = f_diff(point, z_min, delta)
                f_min = np.min(f_diff_delta[i])
                tmp = ((abs(f_diff_delta[i] - f_min))**2).sum()
                D_diff[i] = np.sqrt(1/len(delta)*tmp)
            try:
                z_MLS_index = np.argmax(D_diff)
                irradiated_mask[y_m, x_m]['delta_MLS'] = delta[np.argmin(
                irradiated_mask[y_m, x_m]['delta_stab'] = D_diff[z_MLS_in
            except:
                irradiated_mask[y_m, x_m]['flag'] = False
                #pass
```

[22]
```
radiated_mask['delta_stab'] = np.abs(irradiated_mask['delta_MLS'])*irradia
```

[23]
```
delta_MLS = []
for x_m in range(irradiated_mask.shape[1]):
    for y_m in range(irradiated_mask.shape[0]):
        if irradiated_mask[y_m, x_m]['flag']:
            delta_MLS.append(irradiated_mask[y_m, x_m]['delta_MLS'])
```

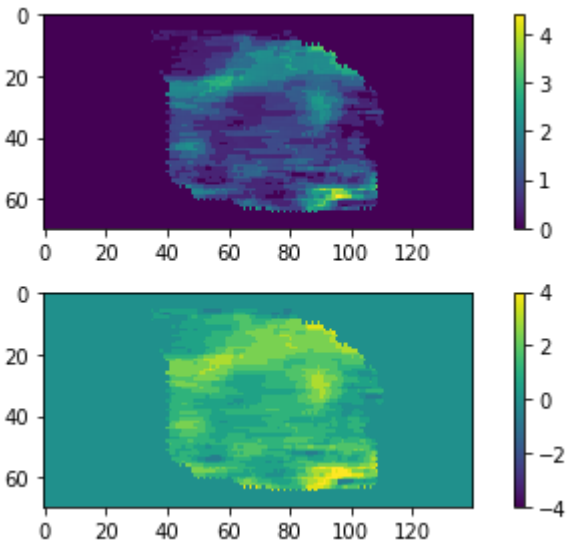
[24]
```
count, edges, _ = plt.hist(delta_MLS, bins = int((np.max(delta_MLS)-np.mi
print('mean :{} std:{}'.format(np.mean(delta_MLS), np.std(delta_MLS)))
print('max count : {}, mode : {}'.format(max(count), ((edges[1:]+edges[:-
```

```
mean :1.367754340171814 std:0.8636655807495117
max count : 1685.0, mode : 1.5
```



[30]
```
plt.subplot(2, 1, 1)
plt.imshow(irradiated_mask['delta_stab'])
```

```
    plt.colorbar()
    plt.subplot(2, 1, 2)
    plt.imshow(irradiated_mask['delta_MLS'], vmin= -4, vmax = 4)   #cmap='RdBu
    plt.colorbar()
    plt.tight_layout()
```



[26]
```
irradiated_mask['delta_MLS'].astype(np.float32).tofile(os.path.join(patien
```

[27]
```
from scipy import stats
a = stats.zscore(irradiated_mask['delta_MLS'][irradiated_mask['flag']==Tr
a.sum()/len(a)
```

```
0.2129424778761062
```

[28]