Techniki obrazowania medycznego - schemat rozwiązania
Anastazja Kąkol, Alicja Kojs, Marcin Oleś, Wojciech Rolka

Na początku należy pobrać i załadować wszystkie dane. Można to otrzymać za pomocą komend:

```
! curl -s
https://packagecloud.io/install/repositories/github/git-lfs/script.deb.
sh | sudo bash
! sudo apt-get install git-lfs
! git lfs install
! git clone https://github.com/neheller/kits19.git
%cd kits19
! python -m starter_code.get_imaging
```

Aby sprawdzić poprawność kodu pobrano jeden pomiar, na którym będą przeprowadzane dalsze przekształcenia:

```
from starter_code.utils import load_case
volume, segmentation = load_case("case_00123")
```

albo skorzystać z funkcji:

```
def load_scan(path):
    slices = [dicom.read_file(path + '/' + s) for s in
os.listdir(path)]
    slices.sort(key = lambda x: int(x.InstanceNumber))
    try:
        slice_thickness = np.abs(slices[0].ImagePositionPatient[2] -
slices[1].ImagePositionPatient[2])
    except:
        slice_thickness = np.abs(slices[0].SliceLocation -
slices[1].SliceLocation)
    for s in slices:
        s.SliceThickness = slice_thickness
    return slices
```

Kolejnym krokiem jest szereg przekształceń dokonywanych na obrazie, który wczytaliśmy. Kolejno trzeba wykonać: przekształcenie obrazu na obraz binarny, filtrację obrazu przy użyciu maski, binearyzację obrazu przefiltrowanego, erozję, indeksację i dylatację obrazu oraz porównanie powstałych obrazów. Można to uczynić korzystając ze stworzonej funkcji :

```
def getBinaryImage(img, threshold, mode='upper'):
    if mode == 'upper':
        return im_gray > threshold
    elif mode == 'lower':
        return im_gray < threshold
```

```
    else:
        raise Exception("Mode not supported")

mask = np.array([[1,2,1],[0,0,0],[-1,-2,-1]])
display(mask)
im_filtered = signal.convolve2d(im_gray, mask)
eroded = morphology.erosion(thresh_img,np.ones([3,3]))
dilation = morphology.dilation(eroded,np.ones([8,8]))
labels = measure.label(dilation)
label_vals = np.unique(labels)
regions = measure.regionprops(labels)
good_labels = []
for prop in regions:
      B = prop.bbox
      if B[2]-B[0]<row_size/10*9 and B[3]-B[1]<col_size/10*9 and
B[0]>row_size/5 and B[2]<col_size/5*4:
            good_labels.append(prop.label)
mask = np.ndarray([row_size,col_size],dtype=np.int8)
mask[:] = 0
for N in good_labels:
        mask = mask + np.where(labels==N,1,0)
    mask = morphology.dilation(mask,np.ones([10,10]))
def compareImages(img1, img2):
    plt.figure(figsize=(12,12))
    plt.subplot(1, 3, 1)
    plt.imshow(img1, cmap='gray')
    plt.title('original')
    plt.axis('off')
    plt.subplot(1, 3, 2)
    plt.imshow(img2, cmap='gray')
    plt.title('processed')
    plt.axis('off')
    plt.subplot(1, 3, 3)
    if (img1.dtype == bool and img2.dtype == bool):
        plt.imshow(np.asarray(img2, dtype=np.uint16) - np.asarray(img1,
dtype=np.uint16), cmap='gray')
    else:
        plt.imshow(img2 - img1, cmap='gray')
    plt.title('difference')
plt.axis('off')
```