Project 2

Segmentation and analysis of pelvic bone in CT images

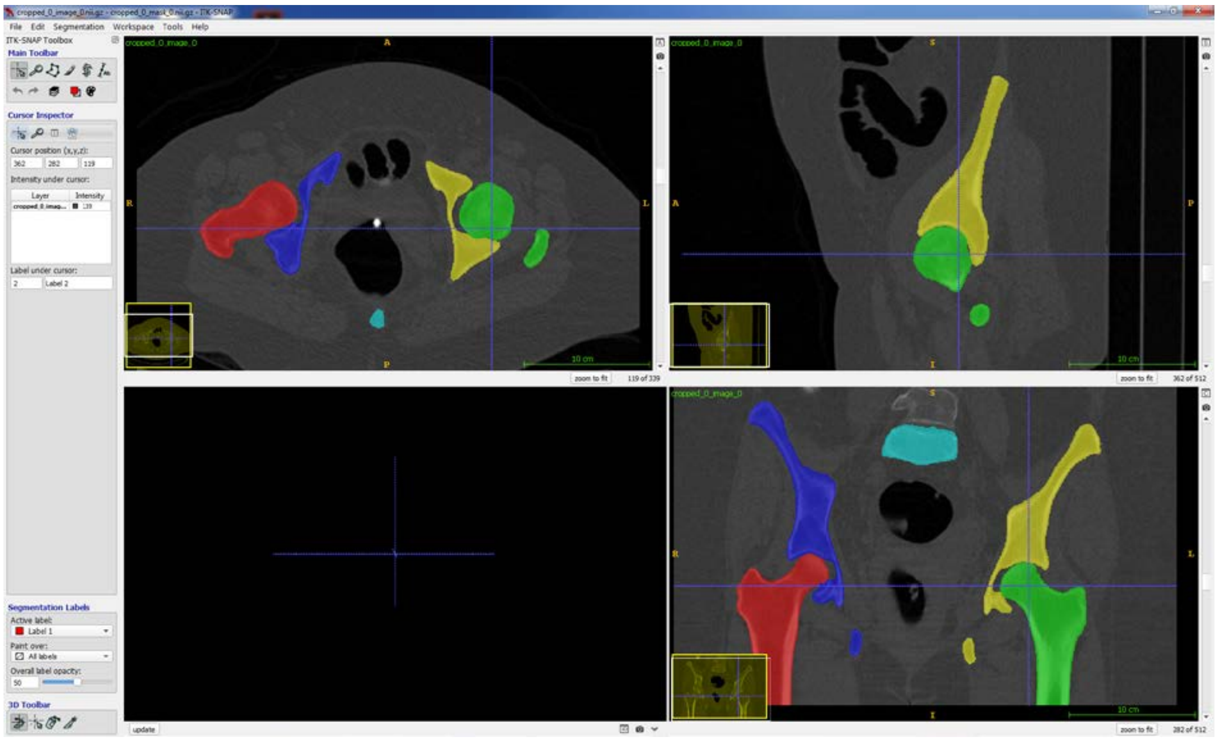
Introduction:

In hip surgery planning, it is important to segment the pelvic bone and the femur. One appealing strategy in case of the pelvic bone is to use one or more atlases that can be used for atlas-based segmentation. The aim of this project is to create the tools to perform this task.

Data

Each group of students will have access to **two** datasets of 3 CT images each. The first one is specific for every group while the second one is common for all groups. Additionally, the manually marked ground truth segmentation of the common dataset is provided. These segmentations were performed by an expert radiologist using **5 different categories** (cf. the figure below):

a. Right femur (in red)
b. Left femur (in green)
c. Right hip bone (in blue)
d. Left hip bone (in yellow)
e. Sacrum (in cyan)



In this project we are interested in segmenting the right and left hip bone only, i.e., categories 3 and 4 (the blue and yellow structures in the figure). These ground truths can be used to test the accuracy of your segmentations.

The data is available in the following link:
https://kth.box.com/s/cwgkrll85z9n59nsye13er1sipzppav3
The structure of the directories in this link is as follows:

- **"COMMON_images_mask"**: contains the 3 images that are going to be used by all groups ("common_XX_image.nii") and their corresponding segmentations ("common_XX_mask.nii") where "XX" is an internal id.
- **"GROUP_images"**: contains the images to be used by every group of students. The file name of the images has the structure "gXX_YY_image.nii" where XX correspond to the group number and YY is an internal id.

The CT scans and the segmentations that will be used in the project are in NIfTI format (.nii). These images can be opened with the tools we used in the labs. The two datasets will be referred to as *"common"* and *"group"* datasets in the rest of this document

Tasks:

1. **Familiarization**:
   Load an image and the corresponding ground truth e.g. in ITK-snap in order to get familiar with the anatomy of the pelvis in CT images.
2. **Atlas creation**:
   Use **one** of the **techniques** from the course **for segmenting** the **left and right hip bone** in the **6 images,** i.e., the two datasets. You can use **any tool** for this, e.g. code from the labs, *MiaLab*, etc. Apply **manual corrections** for small errors in the segmentations from "*group*", so they can be used as atlases, that is, they have to be as accurate as possible. You **DO NOT** need to do that for the images in "*common*", the expert radiologist already did that already for you!
3. **Registration**:
   Implement **two functions for registration** of the CT images. Notice that you might need both **linear and non-linear registration** in the next steps.
   Hint - Consider using the function signatures (see below): *est_lin_transf, est_nl_transf,* for estimation of linear and non-linear transformations and *apply_lin_transf, apply_nl_transf* for applying the estimated transformations to other images.
4. **Atlas based-segmentation**:
   Using the **segmentations** from the specific dataset (from "*group*") **as atlases**, perform atlas-based segmentation for the images in "*common*". For that, you can follow the following steps:
   a. Take **one image** from "*common*" **as the reference** image and **register each of** the **CT images** in "*group*" to that reference.
   b. **Apply** the **transformations estimated** in the previous step to the segmentations of "*group*" estimated in *task 2*. In this way, you will get **3 different segmentations of** the **chosen reference** image. Combine the 3 segmentations by using **majority voting**, i.e. a voxel belongs to the segmentation mask if it appears in at least 2 of the 3 segmentations.
   c. Follow the same procedure for the other images in "*common*".

   Hint – Consider using the function signature *seg_atlas* (see below) that performs atlas-based segmentation of a single image. Then you can call that function in a loop on all images that you want to segment.
5. **Accuracy assessment**:
   Using the segmentations in the "**COMMON_images_mask**" directory as a reference, **compute** the **Dice coefficient** and the **Hausdorff distance** for the results of *tasks 2* and *4* for the images in *"common"*. **Compare** both **results** and **discuss** them in the report.
   Hint – Use a loop to iterate over the list of output files from *tasks 2* and *4*. In the loop body you can call the functions for computing *Dice coefficient* and *Hausdorff distance* on the particular files. Notice that these measurements are available in most libraries (e.g. SimpleITK).

6. **Image classification**:
   Localizing the femur head is crucial in hip surgery planning. The aim of this task is to **detect** the particular **slice** of the CT image which is **most likely to contain** the **femur head**. The next five steps describe the corresponding algorithm for that purpose:
   a. Consider **each individual axial 2D slice** of the CT images in *"group"* and **manually classify** them as **"containing femur head"** or not. For this, create a binary vector of size **n,** where **n** is the number of slices in the **z** axis of the image. Fill the vector with "0" or "1" depending on if the femur head is visible or not in the specific 2D slice. Do the same procedure for three CT images in *"group"*.
   b. **Train** a **classifier** with these manually created labels. Consider using the function signature "train_classifier" (see below).
   c. Take a test image from *"common"* and use the trained classifier to **select** the slice with the **highest estimated probability** of containing the femur head. For this, use the trained classifier to **obtain probability estimates** of whether or not every single axial 2D slice of the image contains the femur head or not. The slice with the **highest estimated probability** will be the one with the **maximum** score. Consider using the function signature "femur_head_selection" (see below). **Create** a plot with the evolution of the **estimated probability** with respect to **z**.
   d. Follow the same strategy for another two test images in *"common"*.

Bonus: Perform the tasks **2-5** for the femur and/or the sacrum. Compare the results with the hip bones obtained in task **5**.

Tips:

- If you have problems with the size of the images, you can work with downsampled versions of them.
- As you noticed in the labs, non-linear registration tends to be very time consuming. You can start with affine registration and once everything is running, you can work with linear + non-linear.

Assessment:

The project work can be done in groups of **2 people**. Each group has to hand in a report of **maximum 6 pages** describing

1. underlying **problems**
2. basic **theories**
3. solution **strategies** used
4. findings from the **experiments**
5. final **conclusions**.

Make sure to **include contents for each** of the given **tasks**. The highlighted parts may help to put emphasis on certain aspects.

Each group will do a **demo and present** its **results** to the rest of class the **30th of May**. Each group will have **7 min** for this. The report and code must be submitted not later than the **28th of May**. The grades of the mini-exams of the modules *Registration, Segmentation* and *Analysis* are part of the final assessment for this project:

Report:	40 points
Presentation:	20 points
Code:	20 points
Mini exams:	20 points

Good luck!!!

Function signatures

As guidance we provide proposals for function signatures that you might incorporate in your implementation when working on the tasks above. These are intended to help you structure your code but note that you are entirely free in your implementations.

```
def est_lin_transf(im_ref, im_mov):
    """
    Estimate linear transform to align `im_mov` to `im_ref` and
return the transform parameters.
    """
    pass

def est_nl_transf(im_ref, im_mov):
    """
    Estimate non-linear transform to align `im_mov` to `im_ref` and
return the transform parameters.
    """
    Pass

def apply_lin_transf(im_mov, lin_xfm):
    """
    Apply given linear transform `lin_xfm` to `im_mov` and return
the transformed image.
    """
    pass

def apply_nl_transf(im_mov, nl_xfm):
    """
    Apply given non-linear transform `nl_xfm` to `im_mov` and return
the transformed image.
    """
    pass

def seg_atlas(im, atlas_ct_list, atlas_seg_list):
    """
    Apply atlas-based segmentation of `im` using the list of CT
images in `atlas_ct_list` and the corresponding segmentation masks
in `atlas_seg_list`. Return the resulting segmentation mask after
majority voting.
    """
    pass

def train_classifier(im_list, labels_list):
    """
    Receive a list of images `im_list` and a list of vectors (one
per image) with the labels 0 or 1 depending on the axial 2D slice
contains or not the femur head. Returns the trained classifier.

   """
    pass

def femur_head_selection(im, classifier):
   """
    Receive a CT image and the trained classifier. Returns the axial
slice number with the maximum probability of containing a femur
head.

   """
    pass
```