Help & Documentation*

Codes of this library is written by of Image Analysis Group members. Each function includes the programmer's name.

1 IO - Reading Data

1. `load_dcm_profiles(folder_path)` takes the path of a directory, as input, which contains all the folders, where each folder contains all the files corresponding to the images taken at a given time with different depths. It returns a `list` that contains all the data we have. [data includes CT-scan images, name of patient, age, name of hospital, etc.]

2 Core

In this document the terms layer, depth and slice location are used interchangeably.

2.1 Get Image Matrices Out of Profile Data

1. `extract_sliceLocation_names(all_profile, no_slices=10)`

   input:

   - `all_profile` List of all profiles which is output of `load_dcm_profiled(.)`
   - `no_slices` Number of different spacial images of a given liver.

   output: Ordered list of ID's of slices.

   Each profile read off the disk has different information in it like patient name, date of images taken, etc. One of the items stored in a profile is ID of an slice, e.g. -140, -165, etc. The variable corresponding to slice ID is called

*Name of the programmer of functions is written in the code for debugging purposes. If you see any flaw, please contact the programmer.

"SliceLocation" and is reachable as follows: "profile.SliceLocation" for a given "profile".

Number of slices in our data is 10. Hence, the function uses no_slices=10 as a default if it is not provided by the user.

2. `matrix_of_all_times(all_profiles, depth, no_time_steps, image_dimension)`

   It goes through all data and returns a 3D-matrix of images of the same depth and different times. The size of the matrix is (no_time_steps, image_dimension, image_dimension)

   input:

   - `all_profiles` which is the output of `load_images(.)`
   - `depth`: The target layer (currently we have 10 layers.)
   - `no_time_steps`: Total number of time steps the images are taken. (currently 59).
   - `image_dimension` is the dimension of images we have. (currently 512 x 512). maybe we can skip this, inside the function we can look at one image, extract its size, and use it.

   output: `image_matrix`. 3D matrix of images of a certain layer taken at different times.

2.2 Extracting Submatrices

1. `extract_submatrix_center(image_matrix, center_coor, margin_size)`

   This function takes a 3D matrix of size (times, image_size, image_size) and returns sub-matrix of size (times, sub_image_size, sub_image_size).

   input:

   - `image_matrix` is the matrix of images of the same depth and all times.
   - `center_coor` is index or coordinate of the center pixel of the target sub-matrix. So, if the center entry of the sub-matrix we want is A_{ij}, then its coordinate is [i, j].
   - `margin_size` is the size of the margin at each side of the center pixel. So, for example, if **margin_size** = 1, then we will extract submatrix of size 3x3.

output: 3D sub-matrix of size **(all_time, m, m)** where $m = 1+2\times margin\_size$

2. **extract_submatrix(image_matrices, upper_left, sub_size)**:

   input:

   - **image_matrices** is a 3D matrix of images taken at different times of the same depth.
   - **upper_left** is the coordinate of the upper left pixel of the sub-image we want to look at. The user has to think about the indexing like Python.
   - **sub_size** is the size of the sub-matrix we want to look at. It is assumed the sub-matrix is square.

   output: a matrix of size (time_steps, sub_size, sub_size).

2.3 Aggregation

1. **aggregate_2D(matrix, sub_matrix_dim)** is a function that produces a matrix by non-local means

   input:

   - **matrix** is a square matrix of an image to be reduced.
   - **sub_matrix_dim** is number of rows (= number of columns) of the blocks in the matrix. For example if matrix is of size 36 x 36, and we want to look at tiles of size 2 x 2, there will be 9 blocks of size 2 x 2

   output: a matrix of size **m x m**, where **m = sqrt(no_blocks)**.

2. **aggregate_3D(matrix, sub_matrix_dim)** is a function that takes in the 3D matrix of images of the same depth (depth $\in \{1, 2, \ldots, 10\}$) taken at different times (time $\in \{1, 2, \ldots 59\}$)

3 Other functions

1. `rgb2gray(image)` takes an RGB image as an input and returns its gray scale image using MATLAB's RGB coefficients (0.2989, 0.2989, 0.114).