AortaGeomRecon

Jingyi Lin

Mar 08, 2023

MODULES DOCUMENTATION:

This is the design document for the AortaGeomRecon module, a *3D Slicer* extension to perform *Aorta* segmentation and Aorta geometry reconstruction.

You can find the source code, the installation guide, and the user manual in this GitHub repository.

CHAPTER

ONE

THE STEPS BEFORE APPLYING THE MAIN ALGORITHM

The algorithm works better with the chest volume cropped to a rectangular prism that contains the aorta and parts of the other organs such as the backbone, some blood tissues, and the heart. This can be simply done with *3D Slicer* and its submodule Volume rendering. A more detailed guide can be found on this GitHub repository section.

After cropping the volume which only contains the region of interest, the algorithm needed a set of variables inputs from the user.

The variables inputs are:

1. *Descending Aorta* or *Ascending Aorta* centre coordinate.
2. *Qualified coefficient* larger values lose the condition to accept a *segmented slice*, and vice-versa.
3. An integer indicates the number of slices that the algorithm can skip.

CHAPTER

TWO

THE MAIN IDEAS OF THE ALGORITHM

At the beginning of the algorithm, the user's input on the aorta centre is used to generate the seed slice, 2-dimensional circle image. This seed slice will be used as a reference throughout the workflow of the algorithm. Therefore, changing the input of the centre coordinate could generate a completely different segmentation result.

For each slice starting from the user's selected slice going in the inferior or superior direction

1. The algorithm generates the seed image by using SITK::LabelStatisticsImageFilter.

   Note: A *Label Statistic coefficient* can be used to control the size of the circle seed image. However, this is a constant that is not linked to the UI for the user to select in the current version.

2. The algorithm then creates another image, the Euclidean distance transform of a binary image as the image intensity map, and uses it with SITK::ThresholdSegmentationLevelSetImageFilter to create a segmented slice.
3. The algorithm determines whether to accept the segmented slice or not, based on the number of white pixels on the segmented slice, and the qualified coefficient to control the limit.

   Note: To determine whether a segmented slice is acceptable, different conditions are verified for *Descending Aorta* and *Ascending Aorta*. These conditions check are all involved with the *Qualified coefficient*, which is decided by the user. In simple terms, the larger the *Qualified coefficient*, the looser condition on accepting a segmented slice.

4. If the algorithm accepted this segmented slice, a new centre coordinate is calculated and used as the seed coordinate for segmenting the next slice.
5. When a segmented slice is not acceptable, the algorithm will skip this slice if the number of the skipped slice is less than the integer given by the user. The algorithm will try to replace these skipped slices by reading the overlapped area of the previous and the next slice.

CHAPTER

THREE

THE PSEUDOCODE OF THE ALGORITHM

```
segmented_slice = get_image_segment()
# Any value (grey-scaled) above 0 will be set to a white pixel
new_slice = segmented_slice > PixelValue.black_pixel.value
total_coord, centre = count_pixels(new_slice)

# start is always the index of the original slice
# end could be the index of the superior or the inferior of the aorta
for slice_i in range(start, end, step):
    cur_img_slice = cropped_image[:, :, slice_i]
    segmented_slice = get_image_segment()
    new_slice = segmented_slice > PixelValue.black_pixel.value
    total_coord, centre, seeds = count_pixel(new_slice)
    if is_new_centre_qualified(total_coord):
         processing_image[:, :, slice_i] = new_slice
         prev_centre = centre
         prev_seeds = seeds
```

Note: You can find the definitions of each function within the module below

3.1 AortaGeomReconModule

3.1.1 test package

Submodules

conftest module

conftest.**pytest_addoption**(*parser*)
Add argument parser to pytest, we can pass parameters to pytest.

conftest.**pytest_generate_tests**(*metafunc*)
Convert parser arguments to parameters

test_image_preprocessing module

test_image_preprocessing.**DSC**(*ref_image*, *test_image*)
Calculate the Dice similarity coefficient

Parameters

- **ref_image** (*numpy.ndarrays*) – nda to compare
- **test_image** (*numpy.ndarrays*) – nda to compare

Returns The Dice similarity coefficient of the reference and test image

Return type float

test_image_preprocessing.**get_cropped_volume_image**(*testCase*)
Read the cropped volume

Returns The cropped volume sitk image

Return type SITK

test_image_preprocessing.**mean_absolute_error**(*ref_image*, *test_image*)

test_image_preprocessing.**mean_square_error**(*ref_image*, *test_image*)

test_image_preprocessing.**print_result**(*ref_image*, *test_image*, *seg_type*)

test_image_preprocessing.**read_asc_volume_image**(*testCase*)
Read the segmented ascending and descending aorta volume

Returns The segmented ascending and descending aorta sitk image

Return type SITK

test_image_preprocessing.**read_desc_volume_image**(*testCase*)
Read the segmented descending aorta volume

Returns The segmented descending aorta sitk image

Return type SITK

test_image_preprocessing.**read_final_volume_image**(*testCase*)
Read the final segmented aorta volume

Returns The final segmented aorta sitk image

Return type SITK

test_image_preprocessing.**root_mse**(*ref_image*, *test_image*)

test_image_preprocessing.**test_asc_and_final**(*limit*, *qualifiedCoef*, *lsCoef*, *testCase*, *processing_image=None*)

test_image_preprocessing.**test_compare_asc**(*limit*, *qualifiedCoef*, *lsCoef*, *testCase*, *processing_image=None*)
Read a test cases' partially segmented volume (descending aorta), perform ascending aorta segmentation, and compare the result with the existing ascending aorta volume

Parameters

- **limit** (*float*) – the maximum Dice similarity coefficient difference allowed to pass the test.
- **qualifiedCoef** (*float*) – the qualified coefficient value to process descending aorta segmentation.

- **lsCoef** (*float*) – the factor used to determine the lower and upper threshold for label statistics image filter.
- **testCase** (*int*) – the test case to run the test (0-5).

Returns The Dice similarity coefficient of the reference and test image

Return type float

test_image_preprocessing.**test_compare_des**(*limit*, *qualifiedCoef*, *lsCoef*, *testCase*)
Read a test cases' cropped volume, perform descending aorta segmentation, and compare the result with the existing volume

Parameters

- **limit** (*float*) – the maximum Dice similarity coefficient difference allowed to pass the test.
- **qualifiedCoef** (*float*) – the qualified coefficient value to process descending aorta segmentation.
- **lsCoef** (*float*) – the factor used to determine the lower and upper threshold for label statistics image filter.
- **testCase** (*int*) – the test case to run the test (0-5).

Returns The Dice similarity coefficient of the reference and test image

Return type float

test_image_preprocessing.**test_compare_final_volume**(*limit*, *qualifiedCoef*, *lsCoef*, *testCase*, *processing_image=None*)

test_image_preprocessing.**test_prepared_segmenting_image**(*limit*, *qualifiedCoef*, *lsCoef*, *testCase*)

test_image_preprocessing.**transform_image**(*cropped_image*)

Module contents

3.1.2 AortaGeomReconDisplayModuleLib package

Submodules

AortaGeomReconEnums module

class AortaGeomReconEnums.**PixelValue**(*value*)
Bases: enum.Enum

An enumeration.

black_pixel = 0

white_pixel = 1

class AortaGeomReconEnums.**SegmentDirection**(*value*)
Bases: enum.Enum

Enum type describing the segmentation direction. Superior is where the human head is located. Inferior is where the human feet is located.

Inferior_to_Superior = 2

Superior_to_Inferior = 1

class AortaGeomReconEnums.**SegmentType**(*value*)

Bases: enum.Enum

An enumeration.

ascending_aorta = 2

descending_aorta = 1

is_axial_seg()

Return True if the segmentation type is descending or ascending aorta segmentation, False otherwise.

is_sagittal_seg()

sagittal = 4

sagittal_front = 3

AortaSagitalSegmenter module

class AortaSagitalSegmenter.**AortaSagitalSegmenter**(*qualified_coef*, *processing_image*, *cropped_image*)

Bases: object

This class performs Aorta Sagittal segmentation

__segment_sag(*sliceNum*, *factor*, *size_factor*, *current_size*, *imgSlice*, *axial_seg*, *seg_type*)

begin_segmentation()

AortaSegmenter module

class AortaSegmenter.**AortaSegmenter**(*cropped_image*, *starting_slice*, *aorta_centre*, *num_slice_skipping*, *processing_image*, *seg_type*, *qualified_coef=2.2*, *label_stats_coef=3.5*)

Bases: object

This class is used to perform Descending or Ascending aorta segmentation

__count_pixel_asc(*new_slice*)

Use segmented slice to calculate the number of white pixels, and the new centre based on the segmented result.

Returns

tuple containing: int: The total number of the X coordinates where it is white pixel tupple: The new derived centre calculated by the mean of X coordinates and Y coordinates of white pixels list: More possible coordinates based on the new derived centre

Return type (tuple)

__count_pixel_des(*new_slice*)

Use segmented slice to calculate the number of white pixels, and the new derived centre for descending aorta segmentation.

Returns

tuple containing: int: The total number of the counted X coordinates tupple: The new derived centre calculated by the mean of counted X coordinates and Y coordinates

Return type (tuple)

__filling_missing_slices()
The helper function to replace the missing slice that was not accepted during the descending aorta segmentation. This function will replace the missing slice by reading the previous slice and the next slice, fill the slice with the overlapping area of both slices.

__get_image_segment()
Use SITK::LabelStatisticsImageFilter to derive a 2d circle image (seed). Use SITK::SignedMaurerDistanceMap to calculate the signed squared Euclidean distance transform of the circle. Get segmented image slice based on the seed image.

Returns Segmented image slice based on intensity values.

Return type numpy.ndarray

__is_new_centre_qualified(*total_coord*, *is_overlapping*)
Return True if the number of coordiante in the segmented centre is qualified

Returns Boolean

__is_overlapping(*img1*, *i*)
Compare the current segmented slice with the previous two slices, return True if any overlaps otherwise False

Returns comparison result

Return type Boolean

__prepare_seed()
Get a seed from the original image. We will add extra space and use it to get the labeled image statistics.

Returns An image slice with aorta centre and some extra spacing.

Return type SITK::IMAGE

__segmentation()
From the starting slice to the superior or the inferior, use label statistics to see if a circle can be segmented.

begin_segmentation()
The public method to process segmentation.

property cropped_image

property processing_image

Module contents

CHAPTER

FOUR

INDICES AND TABLES

genindex
modindex

4.1 Glossary of Terms Used in AortaGeomRecon Documentation

Aorta The aorta is the largest artery of the body and carries blood from the heart to the circulatory system. It has several sections: The Aortic Root, the transition point where blood first exits the heart, functions as the water main of the body.

Ascending Aorta The ascending aorta is the first part of the aorta, which is the largest blood vessel in your body. It comes out of your heart and pumps blood through the aortic arch and into the descending aorta.

Descending Aorta The descending aorta is the longest part of your aorta (the largest artery in your body). It begins after your left subclavian artery branches from your aortic arch, and it extends downward into your belly.

DICOM Digital Imaging and Communications in Medicine (DICOM) is the standard for the communication and management of medical imaging information and related data.

Label Statistic coefficient The coefficient used to get a 2d circle image with SITK::LabelStatisticsImageFilter. Larger values with this coefficient implies a larger range of pixels included from the slice. This could be adjusted based on the area of the aorta center.

Note: Label Statistic coefficient is fixed at 3.5 for now. To let the user choses the values, we need to implement an UI parameter in AortaGeomReconDisplay module (our 3D Slicer extension module), and mapped the value from UI to the logic module.

Organ Segmentation The definition of the organ boundary or the organ segmentation is helpful for orientation and identification of the regions of interests inside the organ during the diagnostic or treatment procedure. Further, it allows the volume estimation of the organ.

Qualified coefficient The coefficient used in deteriming the threshold for LabelStatisticsImageFilter.

slice A 2 dimensional image retrived from a 3 dimensional volume.

segmented A image is processed with only some of the image regions or image objects shown.

3D Slicer 3D Slicer (Slicer) is a free and open source software package for image analysis and scientific visualization. Slicer is used in a variety of medical applications, including autism, multiple sclerosis, systemic lupus erythematosus, prostate cancer, lung cancer, breast cancer, schizophrenia, orthopedic biomechanics, COPD, cardiovascular disease and neurosurgery.

PYTHON MODULE INDEX