CS8395 – FALL 2022 - ITK ASSIGNMENT

Jacob (Jake) J. Watson[1*]

[1]Department of Biomedical Engineering, Vanderbilt University, Nashville, TN

1. INTRODUCTION

The goal of this assignment is to write an algorithm to segment the brain from five given grayscale head images, collectively referred to as I_m. The algorithm must be written in the C++ language using the open-source application development framework ITK (Insight Toolkit). To accomplish this segmentation task, we are provided an additional grayscale image, I_{atlas}, and the binary segmentation for this image, L_{atlas}.

The prescribed operations composing the algorithm to achieve a successful segmentation are:

1. Affinely register the I_{atlas} to I_m, which will create a transformation T_m^{aff}
2. Apply the transformation T_m^{aff} to the image I_{atlas} and labels L_{atlas}, to create $I_{atlas}^{aff,m}$ and $L_{atlas}^{aff,m}$.
3. Deformably register the $I_{atlas}^{aff,m}$ to I_m, which will create a transformation T_m^{def}.
4. Apply the transformation T_m^{def} to the binary image $L_{atlas}^{aff,m}$ to create the segmentation L_m
5. Compute the volume of L_m.

A single '.cxx' file must implement the above steps for the five head images of a single subject. The inputs will be I_m, I_{atlas}, and L_m. The algorithm will output two files $L_{atlas}^{aff,m}$ and L_m , and print the volume of L_m on the terminal.

2. REGISTRATION METHOD

The 'ImageRegistrationMethod' class [1, 2] was implemented for the registration methodology because it allows the user to select a particular transformation at the time of image registration. For this assignment, one registration method can be written using this class, and updated accordingly for the respective transforms for affine and deformable registrations. This was done for program simplicity.

The 'MeanSquaresImageToImageMetric' class [3, 2] was implemented for the metric of the registration method. This metric is intensity-based and attempts to minimize the mean of the sum of squared differences between the fixed image and the moving image.

The 'RegularStepGradientDescentOptimizer' class [4, 2] was implemented for the optimizer of the registration method. Gradient descent is an iterative optimizer that finds a local minimum in a differential function. This optimizes the registration method by moving the moving image in the direction of the local minimum, which is the opposite direction of the gradient at the current point. This optimizer class also includes constraints to prevent steps that are too large; this feature is not available in the 'RegularStepGradient' class.

3. TRANSFORMATIONS AND REGISTRATIONS

An affine transformation is a rigid transformation technique allows linear mapping with the addition of a constant offset, commonly in Euclidean coordinates. An affine transformation is uniformly applied to the image, and is capable of scaling, sheering and mirroring/reflecting across a plane The 'AffineTransform' class [5, 2] was implemented in the registration method to affinely register I_{atlas} to I_m to create a transformation T_m^{aff} for the first operation of the algorithm. This transformation was then applied to the image I_{atlas} and labels L_{atlas} to create $I_{atlas}^{aff,m}$ and $L_{atlas}^{aff,m}$ for the second operation of the algorithm.

A deformable transformation is a non-rigid transformation that allows non-uniform mapping between images. This transform is spatially variant and can change the shape of the object being transformed. This can correct small discrepancies between the fixed and moving images by deforming the moving image to the fixed image. The 'BSplineTransform' class [6, 7, 8, 9] was implemented, which models the deformation field using basis splines, or B-splines. The fixed parameters of the B-spline transform include origin, dimension, direction, and mesh size and are based on the specifications of the input image. The B-spline transform was then uploaded into the registration method to deformably register the $I_{atlas}^{aff,m}$ to I_m to create a transformation T_m^{def} for the third operation of the algorithm. This transform was then applied to the binary image $L_{atlas}^{aff,m}$ to create the segmentation L_m.

The ‘ResampleImageFilter’ class [10, 2] was used to apply both transforms, and the outputs were then binarized using the ‘BinaryThresholdImageFilter’ class [11] in a function [2] to prepare for displaying in ITK.

4. VOLUME COMPUTATION

To do the volume computation, my final implementation attempted to used the ‘StatisticsImageFilter’ [12] class to retrieve the spacing of the segmentation and calculate the volume. Initially, a mesh was implemented to calculate the volume based on the segmentation, however, this proved to difficult to compile.

5 RESULTS, DISCUSSION, AND ALTERNATIVES

Figure 1 is attached in **Supplemental Document 1: Algorithm Results** (second .pdf document) and shows the results of the algorithm. The algorithm is able to compile, and will output affine transformations. However, the algorithm, is unable to successfully perform the deformable registration due to issues with parameter initialization. Because of this, the algorithm is unable to output deformed transformation and calculate the volume of the segmentation. An optimizer was initially implemented [2], but commented out from the code to troubleshoot deformable registration issues. The class ‘BSplineTransformInitializer’ was implemented for troubleshooting, but removed due to continued parameter initialization issues.

6. REFERENCES

[1] “Itk::ImageRegistrationMethod< tfixedimage, tmovingimage > class template reference,” *ITK*. [Online]. Available: https://itk.org/Doxygen/html/classitk_1_1ImageRegistrationMethod.html. [Accessed: 15-Nov-2022].

[2] “VU-CS8395-fall2022,” *GitHub*. [Online]. Available: https://github.com/VU-CS8395-Fall2022. [Accessed: 15-Nov-2022].

[3] “Itk::meansquaresimagetoimagemetric< tfixedimage, tmovingimage > class template reference,” *ITK*. [Online]. Available: https://itk.org/Doxygen/html/classitk_1_1MeanSquaresImageToImageMetric.html. [Accessed: 15-Nov-2022].

[4] “Itk::Regularstepgradientdescentoptimizer class reference,” *ITK*. [Online]. Available: https://itk.org/Doxygen/html/classitk_1_1RegularStepGradientDescentOptimizer.html. [Accessed: 15-Nov-2022].

[5] “Itk::affinetransform< tparametersvaluetype, vdimension > class template reference,” *ITK*. [Online]. Available: https://itk.org/Doxygen/html/classitk_1_1AffineTransform.html#a9927ecbd01394673a9f212a9c55acfbe. [Accessed: 15-Nov-2022].

[6] “Itk::BSplineTransform< tparametersvaluetype, vdimension, vsplineorder > class template reference,” *ITK*. [Online]. Available: https://itk.org/Doxygen/html/classitk_1_1BSplineTransform.html. [Accessed: 15-Nov-2022].

[7]“Examples/registrationitkv4/deformableregistration13.cxx,” *ITK*. [Online]. Available: https://itk.org/Doxygen/html/Examples_2RegistrationITKv4_2DeformableRegistration13_8cxx-example.html#_a13. [Accessed: 15-Nov-2022].

[8] *3 registration*, 28-May-2019. [Online]. Available: https://itk.org/ITKSoftwareGuide/html/Book2/ITKSoftwareGuide-Book2ch3.html. [Accessed: 15-Nov-2022].

[9] “Examples/registrationitkv4/iterativeclosestpoint2.cxx,” *ITK*. [Online]. Available: https://itk.org/Doxygen/html/Examples_2RegistrationITKv4_2IterativeClosestPoint2_8cxx-example.html. [Accessed: 15-Nov-2022].

[10] “Itk::ResampleImageFilter< tinputimage, TOutputImage, TInterpolatorPrecisionType, ttransformprecisiontype > class template reference,” *ITK*. [Online]. Available: https://itk.org/Doxygen/html/classitk_1_1ResampleImageFilter.html. [Accessed: 15-Nov-2022].

[11] “Itk::BinaryThresholdImageFilter< tinputimage, TOutputImage > class template reference,” *ITK*. [Online]. Available: https://itk.org/Doxygen/html/classitk_1_1BinaryThresholdImageFilter.html. [Accessed: 15-Nov-2022].

[12] “Itk::StatisticsImageFilter< tinputimage > class template reference,” *ITK*. [Online]. Available: https://itk.org/Doxygen/html/classitk_1_1StatisticsImageFilter.html. [Accessed: 15-Nov-2022].