Improving Spinal Cord Segmentation and Cross-sectional Area Calculation

Joo-won Kim, Junqian Xu
Icahn School of Medicine at Mount Sinai, New York, NY, USA

Icahn School of Medicine at Mount Sinai
Translational and Molecular Imaging Institute

Introduction

Quantifying cross-sectional areas of the spinal cord accurately is important for monitoring disease progression (i.e., spinal cord atrophy) in neurodegenerative diseases, such as multiple sclerosis or spinal cord injury. Recently, the Spinal Cord Toolbox, developed by Polytechnique Montréal, McGill University, and Aix-Marseille Université, has provided a convenient tool for automated spinal cord segmentation and cross-sectional area quantification. In this study, we estimated a spinal cord inner line and fed it to the Spinal Cord Toolbox segmentation algorithm (i.e., PropSeg) to improve the spinal cord segmentation in non-optimal signal-to-noise ratio (SNR) and/or contrast-to-noise ration (CNR) images. In addition, we introduce a robust method to approximate the tangent vector to the cross section of the spinal cord.

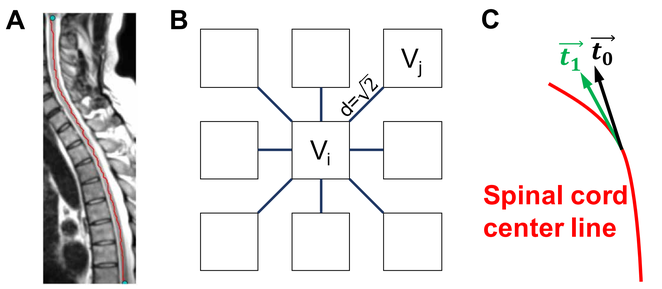

Figure 1. Schematic diagrams of the proposed inner line extraction and varying normal vectors methods. In (A), the red curve is the inner line, calculated from the shortest path between the two cyan color end voxels. In (B), the node V_i has 26 edges and 8 of them are shown. The Euclidean distance between V_i and V_j is square root of 2. In (C), two tangent vector estimation (black and green) are shown on a spinal cord center line (red).

Methods

Inner Line Extraction

The inner line of a spinal cord in an MR image was extracted by finding the shortest path between two voxels (Fig. 1A, cyan color, manually chosen) on a graph G (Fig. 1B). The nodes V in G represent voxels in the image; the edges E of a node V_i are the voxels' 26 neighbor voxels (Fig. 1B, 2D sketch); the edge weight w(i, j) of an edge connecting V_i and V_j is defined as

$$w(i, j) = d(V_i, V_j) * abs(I_i - I_j),$$

where $d(V_i, V_j)$ is the Euclidean distance between V_i and V_j, and I_i is the intensity of V_i.

Spinal Cord Segmentation

The PropSeg in the Spinal Cord Toolbox (version 2.0.6) was applied to segment the spinal cord. To evaluate the segmentation results, we compared the segmentation using either default options or '-init' / '-radius' options versus using '-init-centerline' option with the proposed inner line estimation.

Varying Normal Vectors

A spinal cord cross-sectional area is generally calculated by:
1. Estimating the tangent vector at a spinal cord centerline voxel.
2. Generating a plane whose normal vector is the tangent.
3. Calculating the intersecting area using the surface mesh.

Since the estimated tangent vector is susceptible to imperfections of medical images on discrete voxels (e.g., $\vec{t_0}$ versus $\vec{t_1}$ in Fig.1C), we added intermediate steps to improve this estimation accuracy by varying the tangent vector.

1-a. Build a spherical coordinate system with the tangent vector as zenith-axis.

1-b. Create normal vectors defined by polar and azimuth angles.

In this study, 2,881 vectors were tested using 40 polar and 72 azimuth angles.

Step 2 and 3 were repeated for each vector to estimate the spinal cord cross-sectional area. Among the areas from these vectors, the minimal value is the cross-sectional area.

Implementation

Inner Line Extraction

We implemented the Dijkstra's shortest path algorithm with Python (CPU) and a pseudo-shortest path algorithm with Python Numba (CUDA). The latter algorithm updates all nodes instead of saving/calculating the minimum-distance-node (e.g. priority queue in Dijkstra's algorithm).

Cross-sectional Area Calculation

The CPU Python code was implemented to calculate the cross-sectional area sequentially for the normal vectors. The CUDA code (Python Numba) was implemented for one CUDA node to calculate the area of a normal vector.

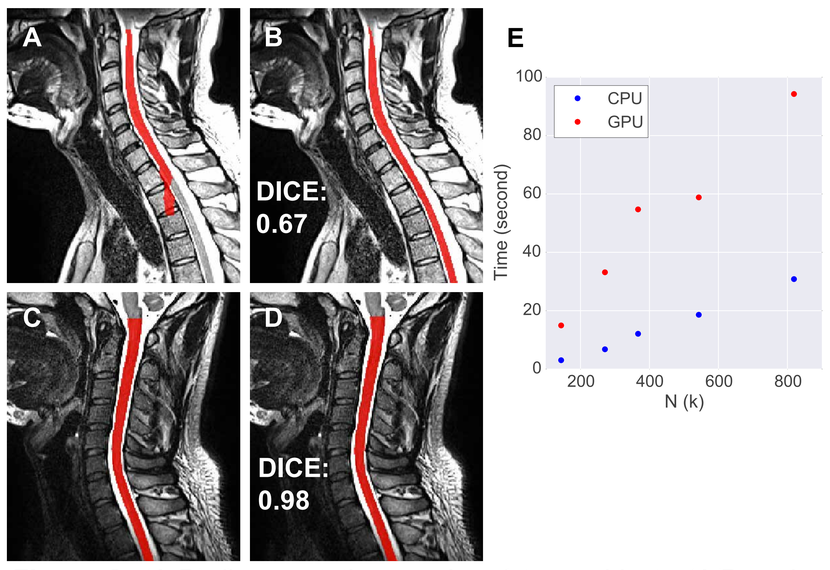

Figure 2. (A-D) Segmentation results of two subjects (A/B and C/D) using default PropSeg options (A, C) or inner lines (B, D). (E) Execution time. In (A-D), red regions are segmentation results. In (E), x-axis indicates the number of voxels, N(k), in the domain.

Results

Five MR images were analyzed using 2.8GHz quad-core Intel Core i7 system with NVIDIA GeForce GT 750M.

Inner Line Extraction

The inner line extraction improved two segmentation results for the cases that the default PropSeg did not segment properly (Fig. 2A,B). For the other three subjects, the PropSeg with or without inner lines performed comparably well (Fig. 2C,D, DICE coefficients > 0.9).

The CUDA implementation was 3X faster than the CPU implementation (Fig. 2E).

Cross-sectional Area Calculation

The cross-sectional areas from multiple normal vectors were up-to 9% smaller than those from a single normal vector (Fig. 3A).

The CUDA implementation was 10X faster than the CPU implementation (Fig. 3B).

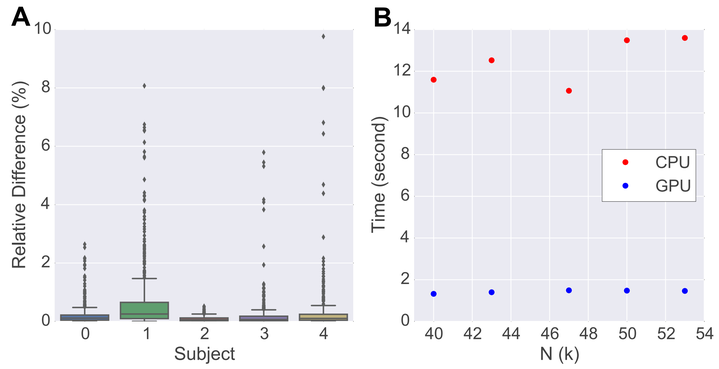

Figure 3. (A) The box plots of relative differences of the cross-sectional areas between a single normal vector and multiple normal vectors at all center line points (~ 200-400 points). (B) execution time. In (B), x-axis indicates the sum of the numbers of nodes, edges, and faces in the surface mesh.

Discussions

The inner line extraction dramatically improved the segmentation result in non-optimal SNR/CNR images.

The multiple normal vectors moderately improved the accuracy of cross-sectional area calculation, considering that the expected area difference between two normal vectors is 1.5% when the angle between the vectors is 10º (cos(10º) = 0.985).

The CUDA implementation of both inner line extraction and cross-sectional area calculation greatly reduced the execution time.

References

• Cohen-Adad et al, Spinal Cord Toolbox: an open-source framework for processing spinal cord MRI data, OHBM 2014
• De Leener et al, Robust, accurate and fast automatic segmentation of the spinal cord, Neuroimage 98, 2014
• Numba, Open Source NumPy-aware optimizing compiler for Python, http://numba.pydata.org/