An Implementation of Parallel Fast Marching Using the Message Passing Interface

Release 0.00

Kevin H. Hobbs[1]

November 23, 2009

[1]hobbsk@ohiou.edu
Biological Sciences
Ohio University
Athens Ohio

Abstract

This document introduces a program based on the algorithm described by Maria Cristina Tugurlan[1]. The program uses file readers, image filters, and file writers from the Insight Toolkit ITK www.itk.org. It produces as output an image whose values are the times of first arrival of a wavefront that spreads from seed points with a speed at every point equal to the input image intensity. It performs the computation in parallel on distributed memory computers using the Message Passing Interface MPI. Each MPI process reads a small piece of the input image into memory. It computes fast marching on its piece. It sends and receives the values from fast marching at piece boundaries. It recalculates fast marching a number of times set from the command line using the new boundary values each time. Each MPI process writes only a small piece of the output file.

A substantial difference is seen when the output of MPI fast marching is compared to the output of serial fast marching. This difference may be acceptable for some uses. The program should be able to handle input images that are too large to fit in the memory of a single computer.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3137]

Contents

1 Disclaimer

This document and the MPI fast marching program accompanying it should not be thought of as a faithful implementation of Tugurlan's algorithm. They represent the best effort of the author to understand and reproduce Tugurlan's work. Since there is a difference between the output of MPI fast marching and serial fast marching there must be an error in this implementation. Suggestions to improve this implementation are encouraged.

2 Introduction

Fast marching is an important algorithm in image processing. Among other things, it is used to identify parts of an image that belong to a particular physical object. One specific use of fast marching is to produce a vector field from confocal images of neurons (Figure 1) that can be used to trace the neuron dendrites from their tips all the way back to the cell body.

A useful metaphor for what fast marching does is the spread of a fire. The input image gives the amount of fuel at each point in space. The brighter each pixel of the input image is, the more fuel there is at that point in space, and the faster the fire will spread through that point in space. The seed points are the points of ignition. The output of fast marching is the time the fire took to reach each point in space.

When run in serial, fast marching maintains lists of points that have already been assigned a time, that will be assigned a time, and points along the boundary between these two lists. In the fire metaphor these lists

are the places that have burned, have not burned, and are burning now. The list of points along the boundary is carefully sorted so that the next point that will be assigned a time is easy to determine.

Maintaining these lists in parallel would be a daunting task, and the amount of communication between MPI processes that would be required would likely limit the speed of MPI fast marching. Tugurlan instead describes an algorithm where each MPI process computes serial fast marching on a piece of the input image. The image pieces overlap by one pixel. The regions where the pieces overlap are used to restart serial fast marching on each MPI process. Fast marching needs to be recomputed for only some of each piece. The process repeats until the results converge to be identical to the output of serial fast marching[1].

3 Details of MPI Fast Marching

This Section describes the implementation of MPI fast marching using ITK. It follows the source code almost block by block. It list the information the program expects to be provided on the command line. It briefly describes how the work and the input image are divided between the MPI processes. It describes how the interfaces between the image pieces are passed between the MPI processes. It describes how the information from the interfaces between pieces is used to restart serial fast marching at each iteration. Finally, it describes how the results from all of the MPI processes are assembled into a single output image file.

3.1 Command Line Arguments

The MPI fast marching program takes four parameters from the command line :

- the input file name (should support streamed reading)
- the seed file name
- the output file name (must support streamed writing)
- the number of iterations

3.2 Image Pieces

Update() is not called on the input file reader. Instead UpdateOutputInformation() is called so that MPI fast marching can read the input image origin, spacing, and region without loading the entire input into memory.

The input image is written to the output file on the first MPI node in streamed pieces immediately after it is read. This is to facilitate writing the output of MPI fast marching to the output file in parallel.

The input image region is split into the same number of pieces as there are MPI processes. The split regions are lengthened by one pixel in the positive direction for each dimension provided that would not create a region outside of the input image. This creates padded regions with one pixel of overlap. Each MPI process stores all of the region splits and padded splits. Each MPI process calculates the overlap of its padded region and all of the other padded regions. These overlap regions are the regions of image data that will be sent to and received from other MPI processes. An image is created for each overlap region to accept data from other MPI processes.

The padded region is extracted from the input image and set as the speed image for serial fast marching. The seed points that are within the piece of the image are set as the initial trial points for serial fast marching.

3.3 MPI Communication

The overlap regions are extracted from the output of serial fast marching and sent to the MPI processes that are working on the neighboring pieces. The same overlap regions are received from the output of serial fast marching on the MPI processes that are working on the neighboring pieces. Care is taken to avoid a dead-lock where both MPI processes are sending or receiving data at the same time.

3.4 Reinitialize Fast Marching

After all of the overlap regions have been received, all of the sent and received overlap regions are compared. If the value at a pixel of the received image is less than the corresponding value of the sent image, then the position and value of the received pixel is set as a new trial point for the next iteration of serial fast marching. The minimum value of these new trial points is recorded.

The minimum value of the new trial points from the overlap regions is used to set the alive points and more trial points for the next iteration of serial fast marching. The whole output of serial fast marching is compared to this minimum value. If the values of a pixel and all of the pixel's neighbor pixels are below the minimum value, then the pixel's position and value are set as alive points (burned) for the next iteration of serial fast marching. If the value of a pixel is below the minimum value, and the value of at least one of the pixel's neighbors is above the minimum value, then the pixel's position and value are added to the trial points (burning now) for the next iteration of serial fast marching.

3.5 Parallel Writing

Finally, the output of the last iteration of serial fast marching from all MPI processes is written in parallel to a single file. Each MPI process extracts its unpadded region from the output of serial fast marching. The extracted region is pasted into the input image. The writer is set to write only the unpadded region of the output of the paste.

4 Serial Fast Marching

When run as a single MPI process with no iterations the output of MPI fast marching is just the output of serial fast marching. Figure 2 shows the output of serial fast marching seeded from three points within two disconnected dendrites.

5 MPI Fast Marching

Figure 3 shows a reversed volume rendering of the output of MPI fast marching run with four MPI processes for eight iterations. The seeds are the same as those used for serial fast marching (Figure 2). There is little apparent difference between the output of serial fast marching and MPI fast marching inside of the dendrites. Therefore it is likely possible to use MPI fast marching for certain tasks.

6 Compare MPI to Serial

Figure 4 shows the difference between the output of serial and MPI fast marching. The differences are greatest outside of the dendrites, particularly near piece boundaries. The fast marching images have values from 0 to about 7000 with values from 0 to about 200 representing the dendrites. The image of the difference between serial and parallel fast marching has values between 0 and 200. The vast majority of the points that are part of the dendrites have a difference between serial and MPI fast marching less than 5. The difference is not decreased with further iterations. One of the tests included with this paper is a direct comparison of the output of serial and MPI fast marching. This test fails. Future versions of MPI fast marching should reduce this error.

However, the small difference in the dendrites between serial and parallel fast marching suggests that this implementation is adequate for at least the task of automatic tracing of confocal microscopy images of florescent dye filled neurons.

7 Software Used

These programs were built using :

- Insight Toolkit (CVS after November 4 2009)
- CMake (CVS)
- VTK (CVS)
- MPI (openmpi-1.2.4)
- Mesa (CVS)

The volume rendering was done off-screen in VTK which was built to use OSMesa.

8 Acknowledgements

This work was done in Scott L. Hooper's lab at Ohio University mailto:hooper@ohio.edu.

This work was funded by the Neuroscience Program at Ohio University.

The original confocal image of the lobster stomatogastric neuron was provided by Jeff Thuma mailto:thuma@ohio.edu.

References

[1] Maria Cristina Tugurlan. *Fast Marching Methods - Parallel Implementation and Analysis*. PhD thesis, Louisiana State University, 29 August 2008. (document), 2

Figure 1: The input image to fast marching is a piece of a confocal image of a lobster stomatogastric neuron. It has been pre-processed with multi-scale vesselness. It is shown in a volume rendering.

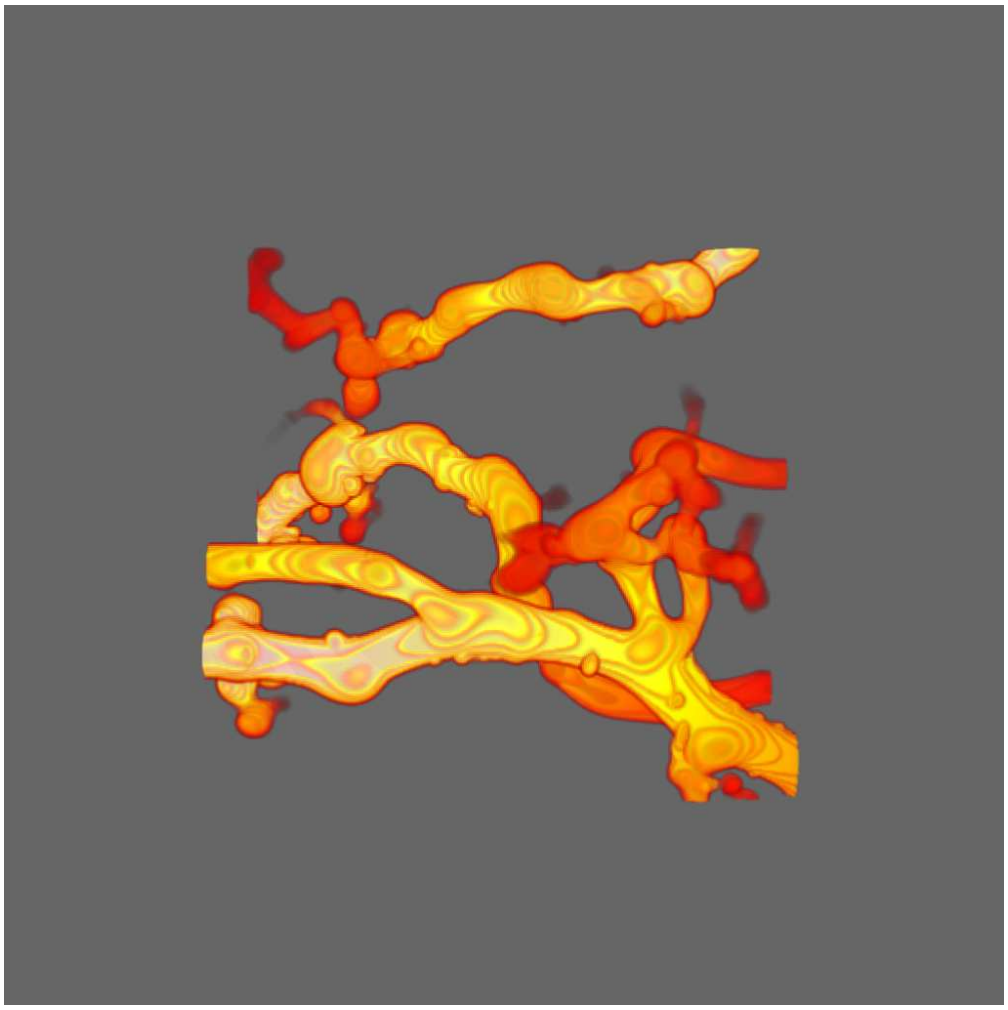

Figure 2: Serial fast marching from 3 seeds on 2 dendrites. Shown as a reversed volume rendering with low values bright and opaque, and high values dark and transparent.

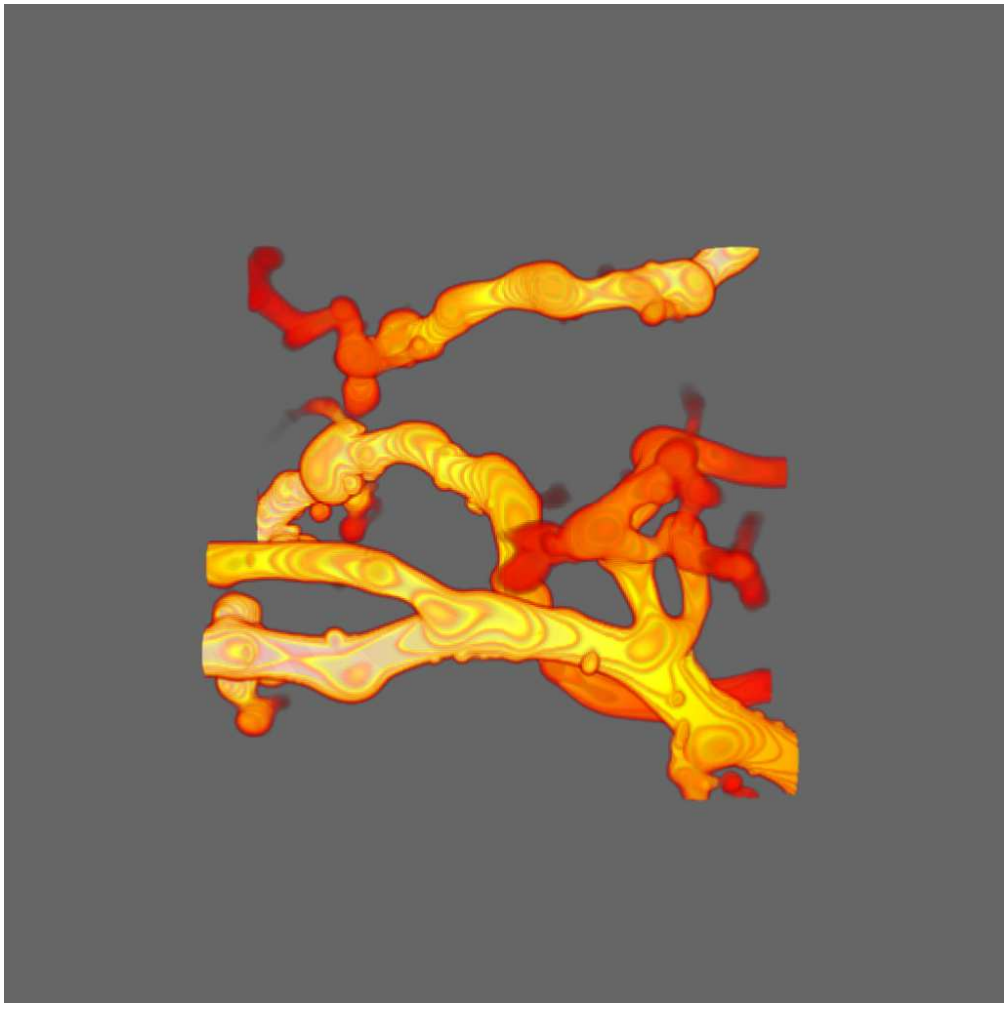

Figure 3: MPI fast marching run with 4 MPI processes for 8 iterations. Shown as a reversed volume rendering.

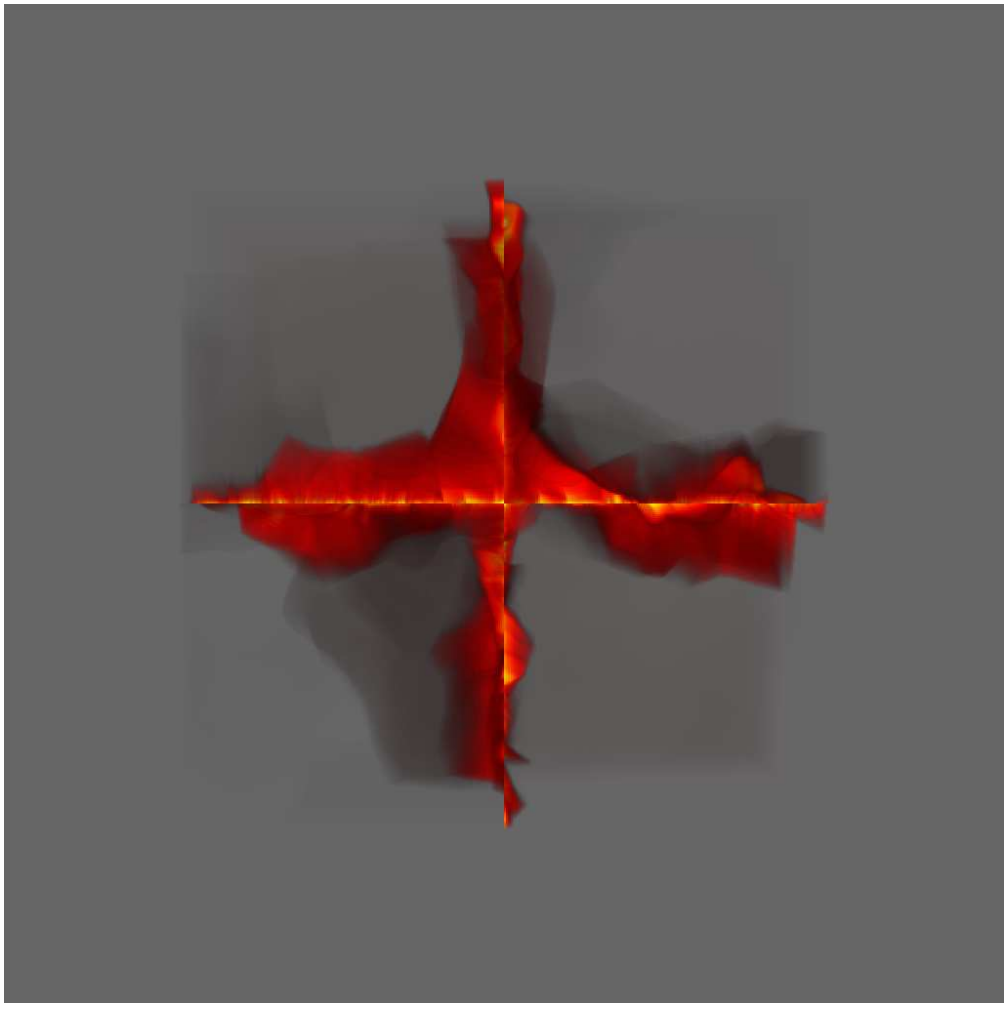

Figure 4: Volume rendering of the difference between MPI and serial fast marching.