

GPU-Based Implementation of a Computational Model of Cerebral Cortex Folding

Jingxin Nie¹, Kaiming^{1,2}, Gang Li¹, Lei Guo¹, Tianming Liu²

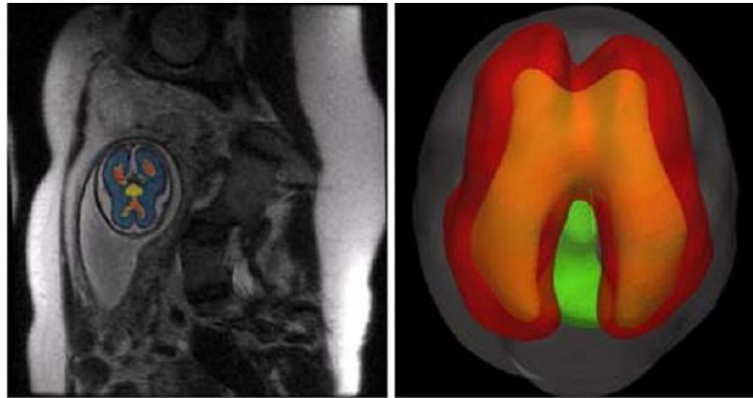
¹ School of Automation, Northwestern Polytechnical University, Xi'an, China,

² Department of Computer Science and Bioimaging Research Center, The University of Georgia, Athens, GA, USA.

Introduction

- Understanding the underlying folding mechanisms has intrigued many people for many years
- To understand the mechanisms underlying cortical folding, computational modeling and simulation is believed to be a very promising approach.
- A GPU-based implementation of a computational 3D geometric model of cerebral cortex folding is more efficient than traditional simulation model.

Materials and Pre-processing



(a)

(b)

- Fetus brain structures (skull, cortex plate, white matter zone, ventricle zone, basal ganglia and thalami) are manually reconstructed from T2 MRI data of fetus brain from 22 to 36 gestation weeks.

Computational model

Cerebral cortex representation

Deformable model of cerebral cortex

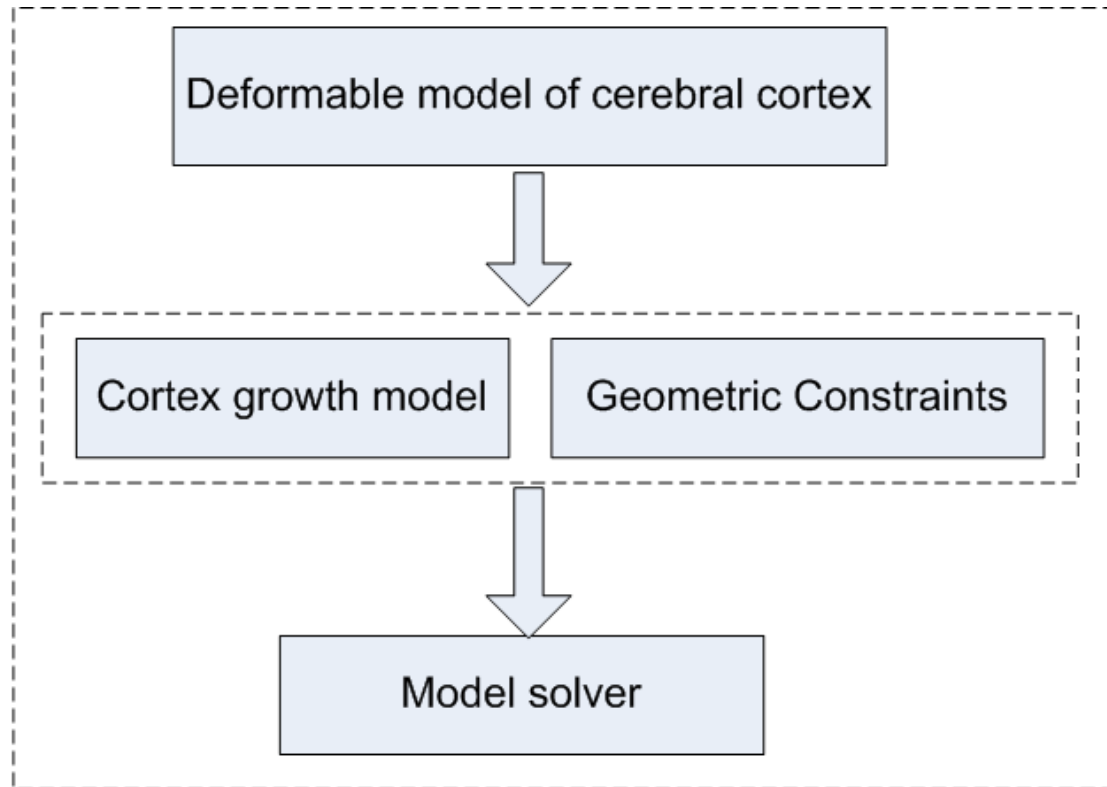
Driving force and constraints

Cortex growth model

Geometric Constraints

Folding simulation

Model solver



Deformable model of the cortex

- The elasto-plasticity model is adopted in our methods.
- The elastic force of each triangle element and the plasticity of cortex are defined along each edge of the triangle.
- The rigidity of cortex is introduced as bending energy.

Cortex Growth Model

- The classic logistic-growth function is adopted to describe the growth of cortical tissues.

Geometric constraints

- A volumetric constraint model is maintained during the simulated folding of the cortex.
- Voxels from the skull, basal ganglia/thalami or ventricular zone are extracted from the scanned 3D MRI image and grouped into a new image as a mask.
- The cortex model can not be deformed into the mask volume.

Model solvers

By combining all equations on the cortical surface together, we have the discrete form of developing cerebral cortex as:

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1} \mathbf{F}(\mathbf{x}, \dot{\mathbf{x}})$$

Where \mathbf{x} is vertex's position, \mathbf{M} is a diagonal mass matrix on vertices, and \mathbf{F} is the net force vector that combines all forces on cortical surface vertices.

Model Solver Iterations

By the Newmark scheme, the dynamic folding progress can be solved by a time-loop, during which each step is described as:

- (1) Grow each element by the classic logistic-growth function at the beginning of each loop.
- (2) Find the stable solution for the growth in step (1) by the following steps iteratively.
 - (a) Apply an adaptation of the reference configuration for each element by plastic property.
 - (b) Calculate elastic forces for each element including bending forces.
 - (c) For each vertex, add all forces from the elements around it.
 - (d) For each vertex, update its new position and velocity.
 - (e) Adjust the new position and velocity, if it does not satisfy the geometric constraints.

GPU-based Implementation

- ***Kernel Arrangement***

There are two types of kernels for element and vertex operation respectively, in which write operations are only performed on element configuration or vertex configuration respectively.

- ***Memory Structure***

All the model variables including element, vertex configurations and the 3D volume for geometric constraints were stored in linear global memory. Because of the unstructured triangulated surface, the access patterns of model variables are random.

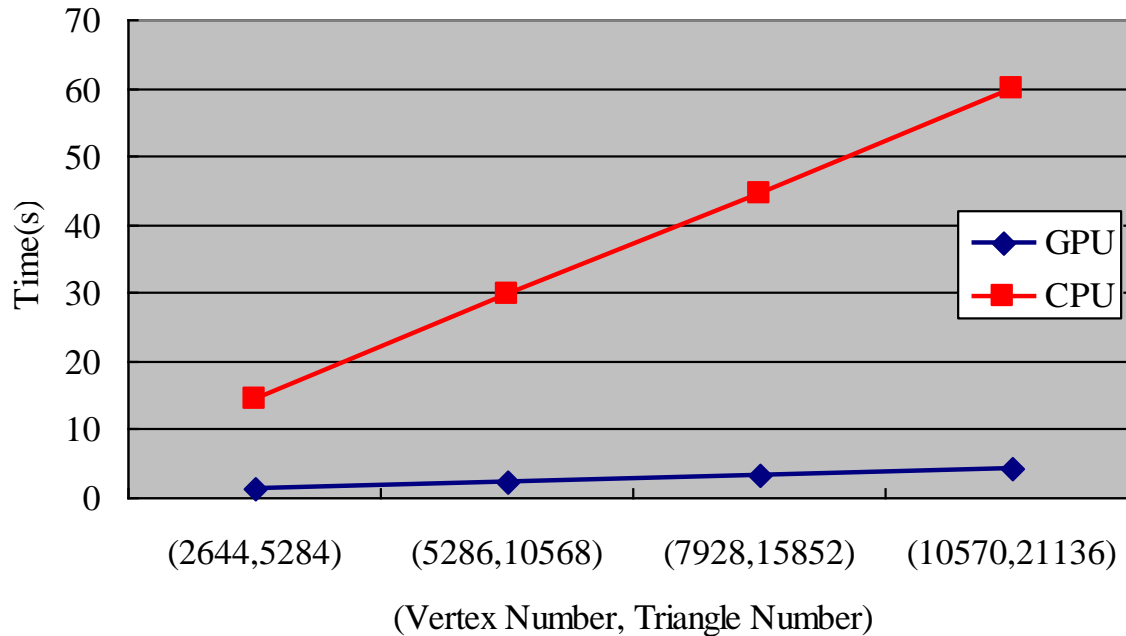
- ***Host-device Interaction***

In our implementation, all the necessary datasets are stored in the device. Also, during the time-loop, no dataset will be transferred from host to device, and the new positions of vertices are transferred back to host for real-time evaluation and visualization.

Results

- Our model is tested on a machine with a single Intel Core2Duo 1.66 GHz CPU, 2 GB of RAM, and a single graphics card NVIDIA GeForce 9600 GT GPU (64 processor core) with a clock speed of 650 MHz and 256 MB of RAM. Both GPU-based and CPU-based implementations are tested for comparison.

Performance comparison between GPU-based vs CPU-based simulations.



Simulation result

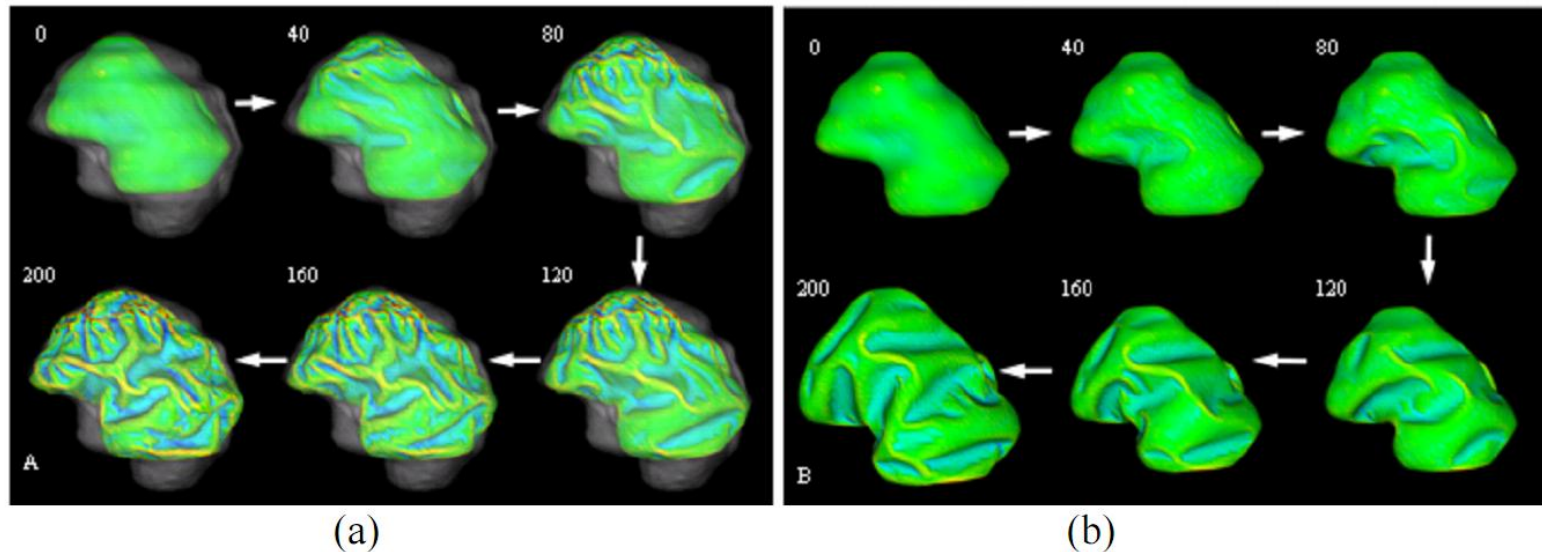



Figure 3. Cortical development with or without skull constraints. (a) and (b) show the snapshots of cortex development at the iteration number 0, 40, 80, 120, 160 and 200 with or without skull constraints respectively. The parameters are set as: $K_c = 1$, $\tau_{ce} = 1000$, $K_b = 5$, $\tau_{cb} = 1000$, $m = 0.002$, $k = 3$. Mean curvature bar: -0.6  0.8.