Evaluation of a Grain Boundary Detection Method Using Cellular Automata

Rundong Jiang, 6/26/19

Estimated total time: 60 minutes

# Learning Outcomes

By the end of this lesson, students will be able to:

* Use cellular automata to detect grain boundaries in a micrograph
* Understand the advantages and limitations of this method
* Enumerate improvement methods based on the specific case

# Audience

College students majoring in materials science and engineering. Some may have prior programming experience, but most don’t.

# Context

This lesson is part of an introduction to computational materials science. Before this lesson, students have already become familiar with the MATLAB environment and just learned the concept of cellular automata. They have also This lesson serves as a hands-on demonstration of how to apply computational methods such as CA to materials science problems.

# Prior Knowledge

* Familiarity with common materials science terminologies such as grain boundary
* Basic understanding of the concept of cellular automata (CA)
* Constructing a CA in MATLAB

# Materials & Preparation

* Jupyter notebook, with parts to be filled in during class
* MATLAB

Before the class, connect laptop to the projector and project the Jupyter notebook on the screen. Make sure at least half of the students have a laptop with a working copy of MATLAB installed and the Jupyter notebook open. Announce to the class that they are welcome to either follow along the Jupyter notebook on their laptop or just watch the projection.

# Motivation (5 min)

Introduce grain boundary detection as an important step for calculating grain size distribution, doing particle analysis, etc. Show students an original micrograph and an image of traced grain boundaries and have students point out the differences. Have students recall the last time they manually traced a micrograph, maybe for an assignment from another MSE class.

# Introducing CA (5 min)

Suggest CA as a way to automatically detect and trace grain boundaries. Ask students to think back to previous instances of CA (e.g. Conway’s Game of Life) and brainstorm how a CA would be constructed and used in this context. Encourage students to approach this problem by relating the properties of a CA (cells, neighborhood, states, rules, # of iterations) to the current problem (isolating certain pixels from the rest in an image).

Then, explain the CA setup (cells, neighborhood, states, rules, # of iterations) and fill in the “basic information” section in the Jupyter notebook.

# CA setup (10 min)

Display the “complete MATLAB implementation” section in the Jupyter notebook. Announce that you will go over the entire code focusing on the parts that pertain to the setup of this particular CA while glossing over the parts that are universal for all CA because students have already seen them. Also announce that they are welcome to either ask questions directly or refer to the “CA setup” section in the Jupyter notebook that is right before the “complete MATLAB implementation” for more detailed explanation, should they have any confusions.

Quickly go over “reading the image file” and “initialization”, focusing on the latter. Let them compare the code with the written setup and make sure they understand the purpose of each line - especially those pertaining to cells, neighborhood, and states - and have them run the code up to “initialization” to make sure the image is loaded into the CA properly.

Then, go over the purpose of the rest of the code, focusing on “updating cell”. Make sure students understand how the written CA rules is translated into a computational context. Let students predict the outcome after one CA iteration and give their reasoning.

# Evaluation (20 min total)

## Evaluation with Test Image

Have students run the entire code on the test image and make sure code works for everyone. Let students compare the CA result with benchmark result from the SUSAN edge detector and discuss 1) whether their prediction of the result is accurate or not, 2) what the CA edge detector did well, 3) what it didn’t do well, and 4) parameters and factors that may have contributed to this result.

## Evaluation with High-Quality and Low-Quality Micrographs

Refer students to the high-quality micrograph and low-quality micrograph in the Jupyter notebook and discuss how their detection results may differ. Have students run the entire code again with both the high-quality micrograph and the low-quality micrograph. Encourage students to experiment with different parameters. Discuss their observations as a class.

# Discussion (10 min)

Encourage students to reflect on CA detection results in three different scenarios (test image, high-quality micrograph and low-quality micrograph) and do a think-pair-share of the advantages, disadvantages, caveats and potential improvements associated with this method. Summarize student opinions and fill in the “discussion” section in the Jupyter notebook as the class shares.

# Conclusion (5 min)

Ask students to summarize what they learned about using CA to detect grain boundaries, then fill in the “conclusion” section in the Jupyter notebook with these sample conclusions, as well as any other takeaways that students generated.

* CA can isolate parts of the grain boundaries, but cannot detect grain boundaries on its own
* CA requires the aid of other image processing techniques in order to achieve better results
* CA has the potential to improve by using more advanced algorithms and optimizing state transition rules

# Celebration & Outroduction (5 min)

Celebrate the learning outcomes that the students have achieved and the fact that the students helped co-construct the Jupyter notebook, and highlight the learnings that weren’t planned and were instead generated by the students.

Encourage students to explore the “further learning” section in the Jupyter notebook in their free time, and thank students for their attention and enthusiasm.