Marcin Szumlański, IS stacjonarne

Multiscale Modelling, 2nd report

Goal of 2nd part of assignment was to implement a grain growth application using Monte Carlo algorithm as foundation. Additional features include dual-phase growth (for both CA and MC interchangeably) as well as MC static recrystallization algorithm – energy distribution, nucleation and growth itself.

Technology:

Java

Java is a general-purpose computer-programming language that is class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

I chose Java due to its ease of use, out-of-the box graphical user interface support (JavaSwing/AWT) and portability. The development time was also important factor.

GUI

The GUI is divided into some sub-parts. First (left-most one) is the CA/MC grain growth, middle one is used for energy distribution purposes whereas the left-most one is used for nucleation + growth. The bottom part is used for dual phase growth for all combinations of CA and MC.

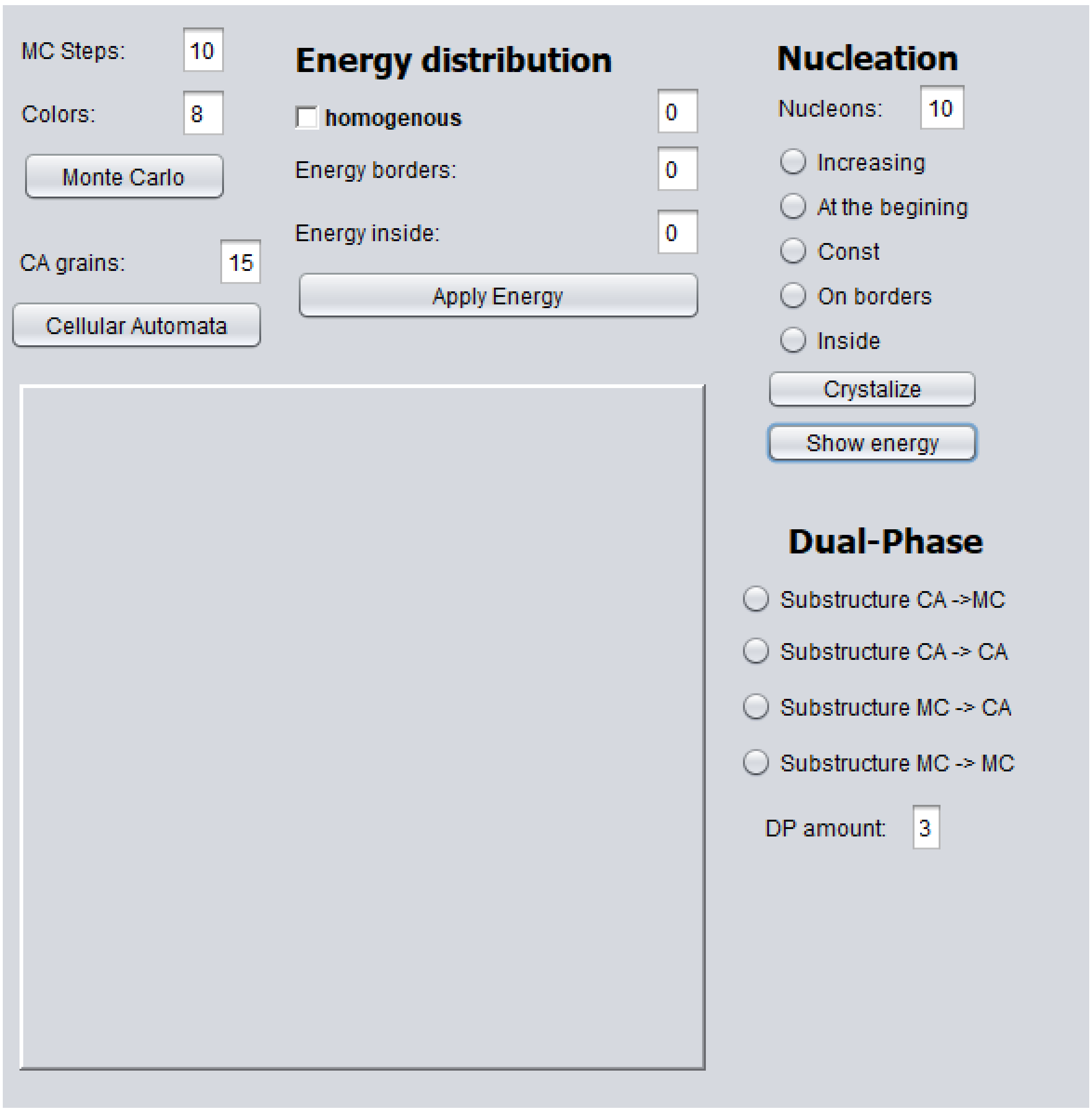

Fig. 1 View of the application

MC steps: amount of full Monte Carlo iterations (all cells have been checked) to run, as well as MC recrystallization algorithm.

Colors: amount of different colors to use during Monte Carlo growth.

CA grains: amount of different grains to randomize over Lattice during CA algorithm

Energy distribution

Homogenous tick-box: allow to switch between homogenous/heterogenous energy distribution.

Energy distribution text fields: how much energy to apply for: homogenous energy distribution (all cells equally) and for heterogenous ones how to divide between borders/insides of generated grains.

Nucleation

Nucleons: amount of different nucleons to seed on Lattice after initial growth is complete.

Radio buttons: switch between 3 different rates of nucleation (constant, increasing and initial) as well as location of nucleons (whether it be borders or insides of grains)

Dual phase

Radio boxes: choose which combination of dual phase growth to proceed with, whether it be CA first and then MC or MC to MC etc.

DP amount: amount of uniquely colored grains to lock in next dual phase.

Implemented classes

All five classes have been implemented – Monte Carlo growth, dual phase for both algorithms, energy distribution, nucleation and recrystallization.

MC growth

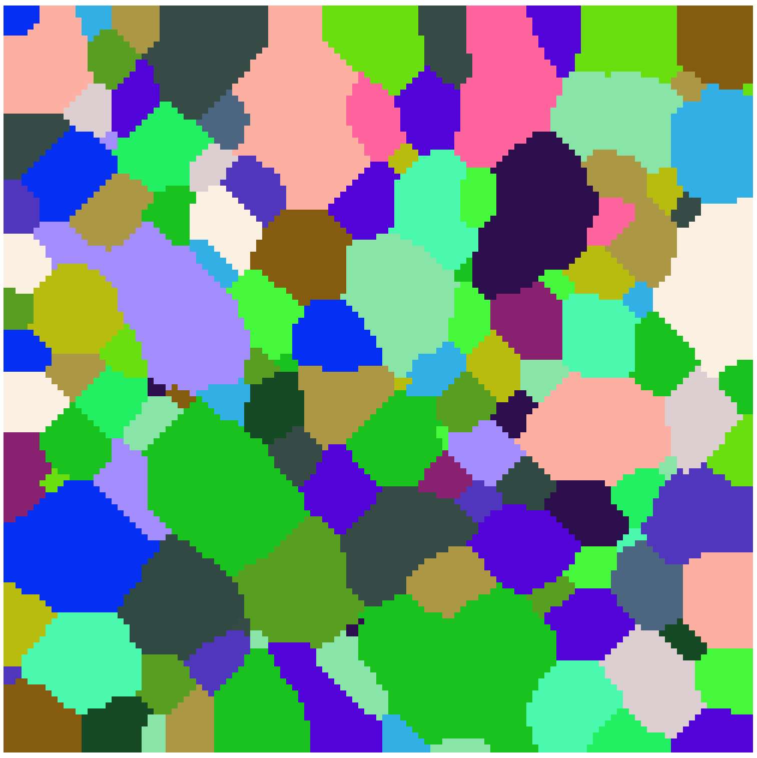

Fig. 2 Monte Carlo grain growth(100 iterations, 10 colors)

Dual Phase

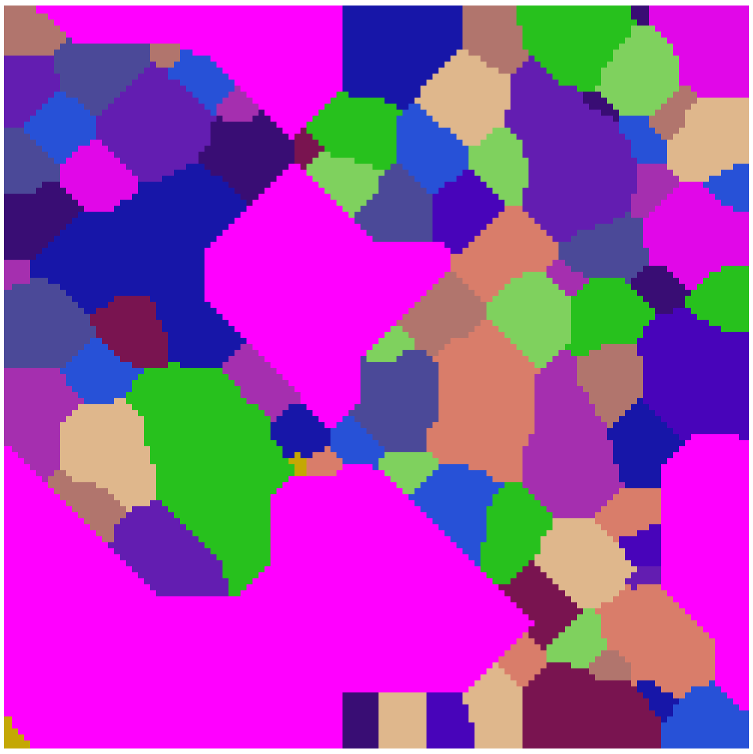

Fig. 3 Dual Phase growth, first CA (10 grains) and after that

Monte Carlo grain growth(100 iterations, 10 colors)

<u>MC recrystallization algorithm</u>

1. Generate grains(Fig. 4) using MC algorithm. In this case I used 10 different colors and 100 MC iterations.

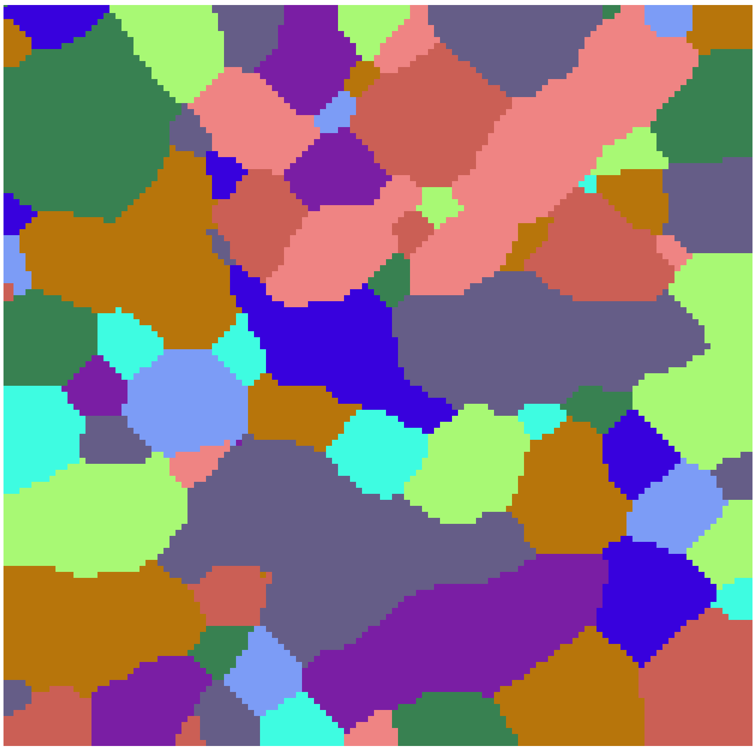

Fig. 4 Generated grains

2. Energy distribution (9 energy for borders and 1 for insides of grains). The Less energy given, the more black the cell, whereas bigger energy the cell color slowly passes to white(Fig. 5).

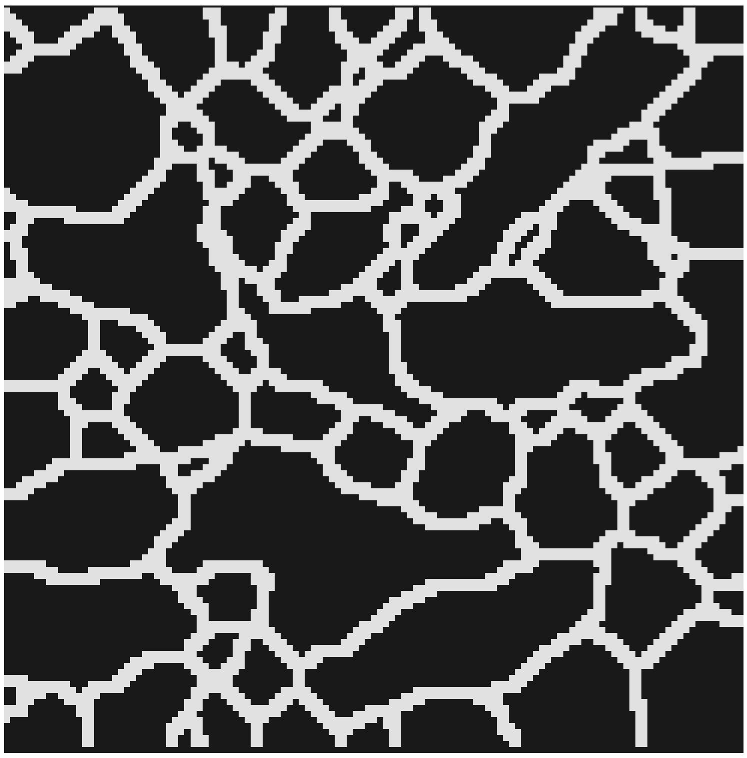

Fig. 5 energy distribution

3. Seed nucleons on borders and grow (in this case I used initial nucleation rate equal to 10)(Fig. 6)

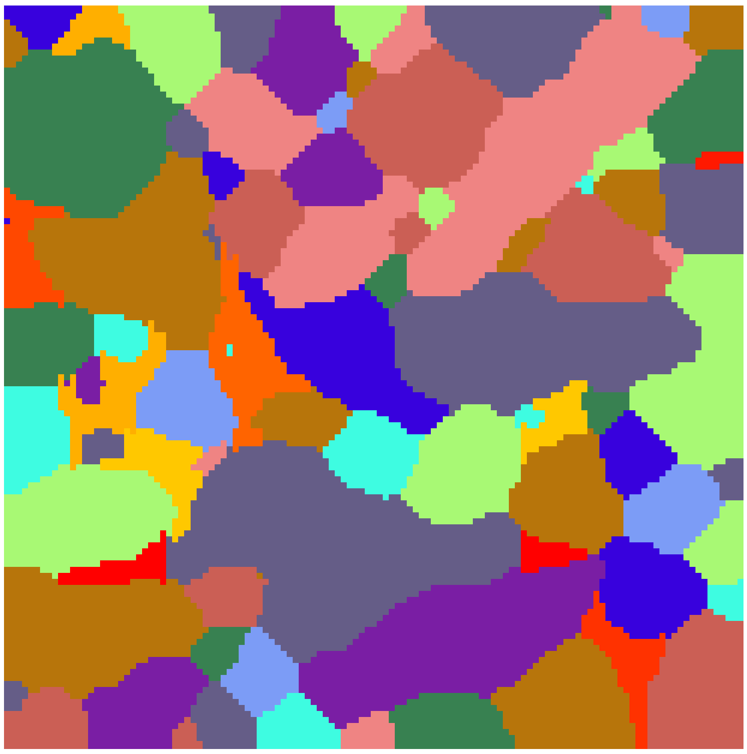

Fig. 6 Nucleation + growth

4. Show energy after recrystallization is complete(Fig. 7).

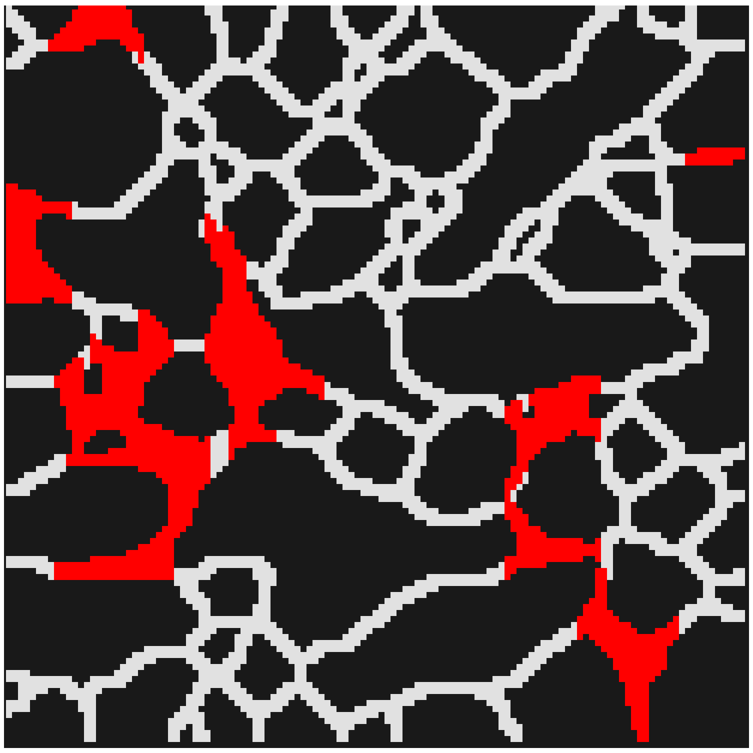

Fig. 7 Energy distribution after recrystallization growth

Comparison of real life vs simulation

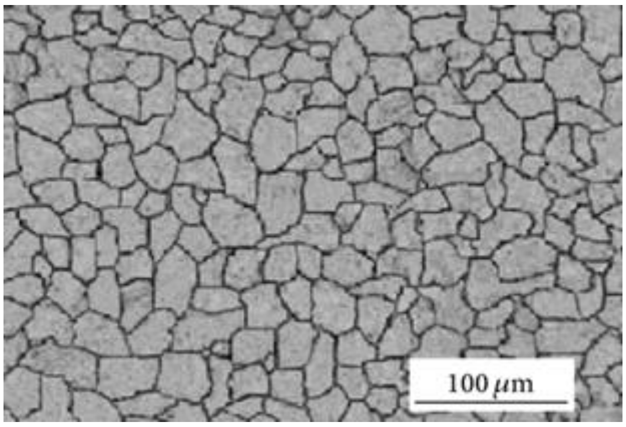

Fig. 8 Microstructure of 316LL austenistic stainless steel

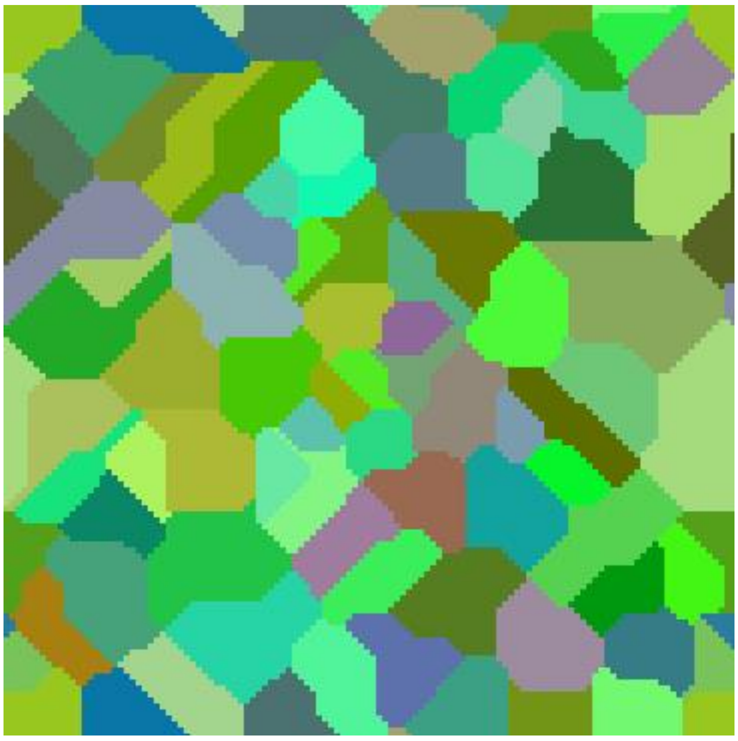

Fig. 9 Generated microstructure using CA(Moore neighbourhood)

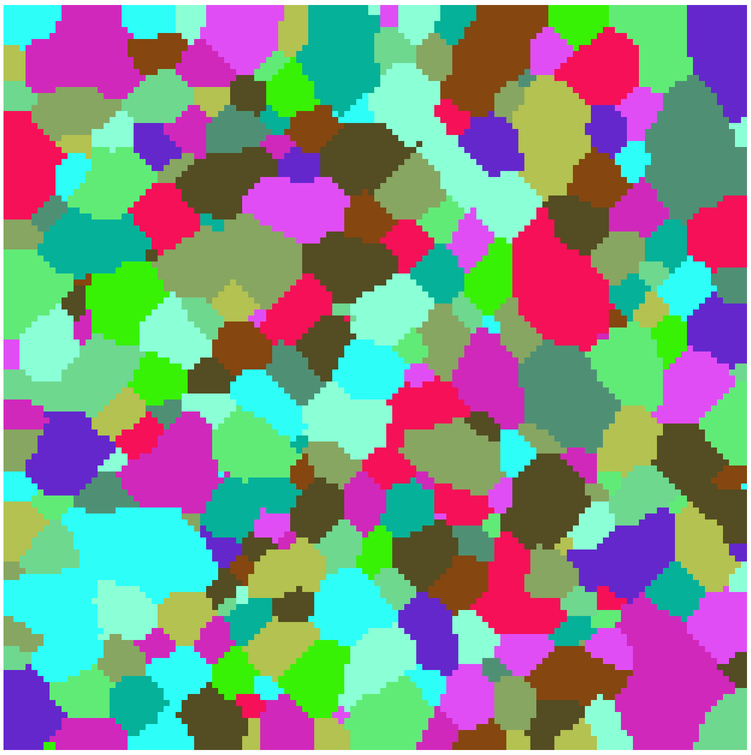

Fig. 10 Generated microstructure using MC

The shape of generated grains is roughly corresponding to the real ones(Fig 8-10).

In the case of CA (I used Moore neighbourhood there) the grains are more “rough” and edges being more straight their shapes are resembling that of polygons(Fig 9).

At the same time the MC generated grains are very round, with no parallel edges(Fig. 10).

If one was to tell, the real-life grain(Fig. 8) is somewhere in the middle. However both simulation do resemble real-life structure pretty accurately.

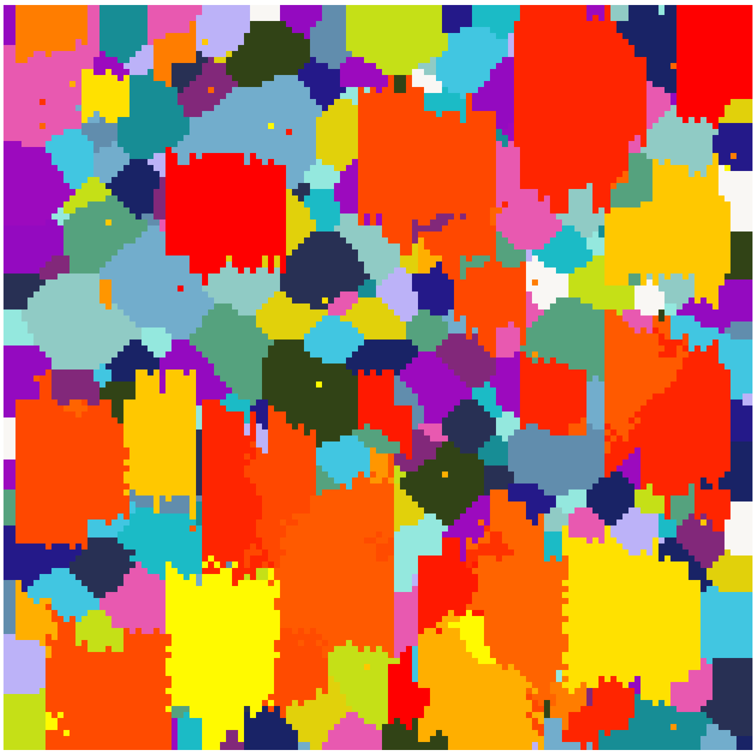

Fig. 11 Generated microstructure using MC static recrystallization algorithm

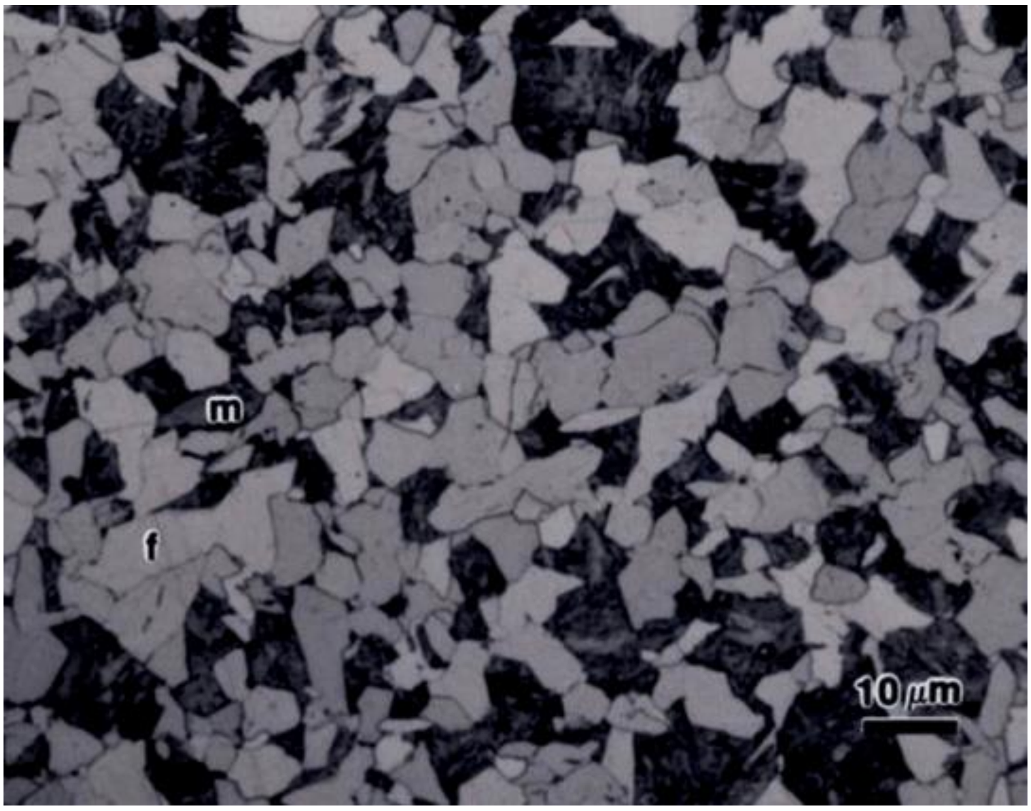

*Fig. 12 Microstructure of 8620 alloy steel austenitized at 927 *C, isothermally transformed ay 677 *C for 1 minute and water quenched after etching*

The generated microstructure(Fig. 11) was created running 50 MC steps + 20 colors, then applying 4 homogenous energy with finally 7 iterations with const nucleation rate in SRX. Using these options it was actually possible to receive similar results as in real-life (Fig. 12) The shapes are again, not perfect similarity but they are really close. However the sizes and locations of grains and recrystallized cells in simulation come from randomness and aren’t super accurate.

Summary

All features required for classes have been implemented including Monte Carlo growth with GUI, dual phase for both algorithms, energy distribution, nucleation and recrystallization.

Java offers a variety of packages useful in such case like Java Swing framework, AWT library, Graphics class and many others.

The generated microstructures do resemble their real life counterparts and look very much alike.

Application runs relatively fast on 125x125 space and its very easy to use, so it can be used for an actual science work regarding studying variety of materials and their characteristics (not always super accurate due to many other factors coming into play when dealing with recrystallization for example).

Most of the performance overhead comes from generating graphic itself. The calculations themselves are much faster. However in my case iteration-wise CA algorithm was a little bit slower than MC, with most performance overhead coming from array copy each iteration. But again, this is implementation dependent and probably can (and should) be implemented differently. However, the CA algorithm yields satisfactory results much faster than MC, due to the fact it only needs to fill its board to complete, whereas MC generation could last for (theoretically) forever, and gives good results after more iteration count than CA.

The SRX generation and its faithfulness to real one is strongly based on nucleation rate and (more importantly) energy distribution. These 2 options need to be fiddled with a lot before yielding accurate results. Also there's the case in our straightforward energy distribution, we only have option to choose between different edge/inside values whereas these values might differ not only because of grain boundaries but also other factors (as in, the energy is not distributed uniformly).

Sources

https://www.researchgate.net/publication/282547287_Microstructure_Strength_and_Fracture_Topography_Relations_in_AISI_316L_Stainless_Steel_as_Seen_through_a_Fractal_Approach_and_the_Hall-Petch_Law [Fig. 8 - Page 4]

https://vacaero.com/information-resources/metallography-with-george-vander-voort/1123-microstructure-of-isothermally-treated-steels.html [Fig. 12]