Multiscale Modelling

Report II

Mateusz Mazur

1. Technology

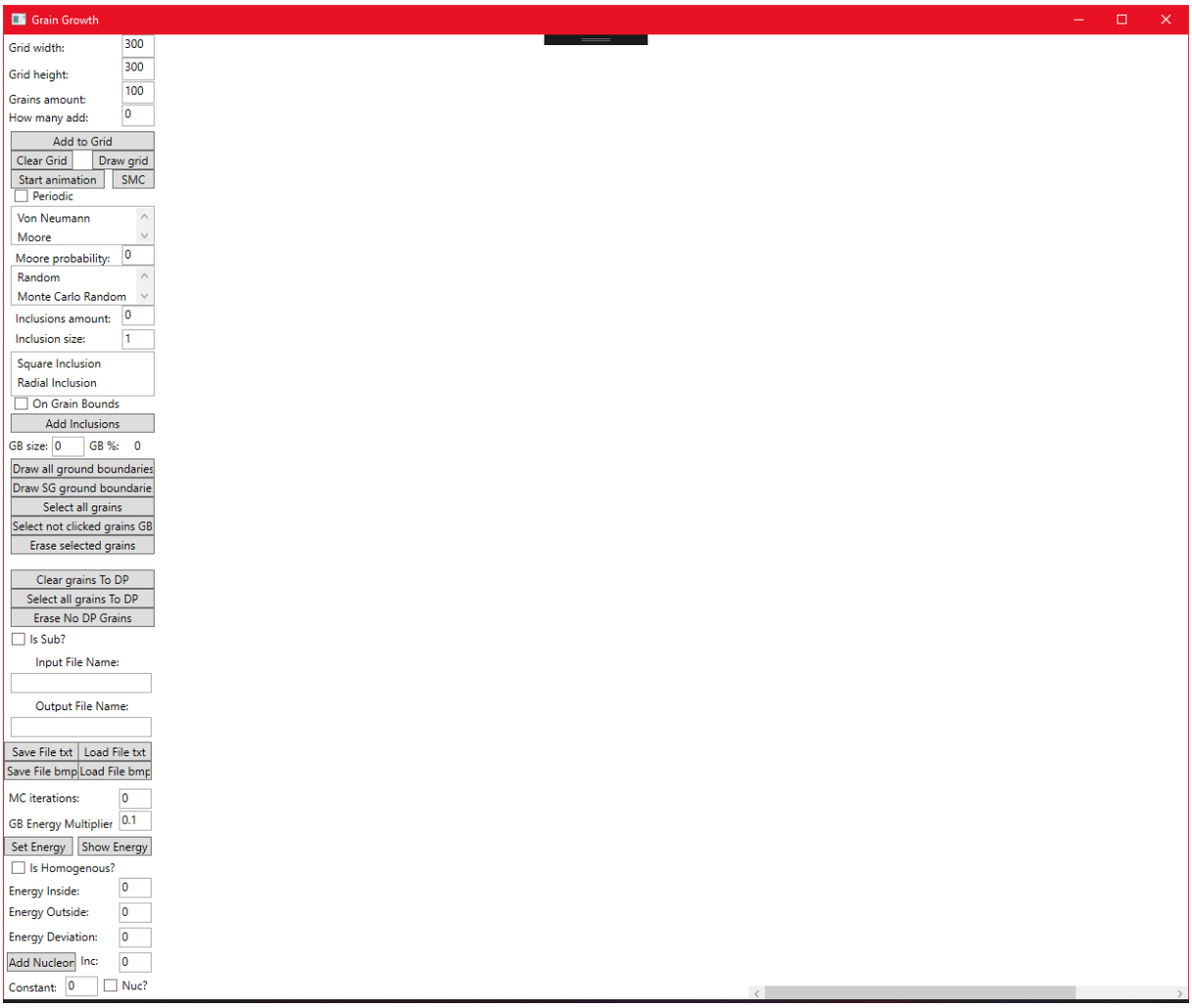
The app is written in C#, which is multi-paradigm programming language encompassing strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented and component-oriented programming disciplines. It was developed around 2000 by Microsoft as part of its .NET initiative. C# is one of the programming languages designed for the Common Language Infrastructure. My app use Windows Presentation Foundation (WPF) which is a graphical subsystem originally developed by Microsoft for rendering user interfaces in Windows-based applications. WPF, previously known as "Avalon", was initially released as part of .NET Framework 3.0 in 2006. WPF uses DirectX and attempts to provide a consistent programming model for building applications. It separates the user interface from business logic, and resembles similar XML-oriented object models, such as those implemented in XUL and SVG.

2. Programme features

This chapter will discuss the main functionalities of the application.

2.1. GUI

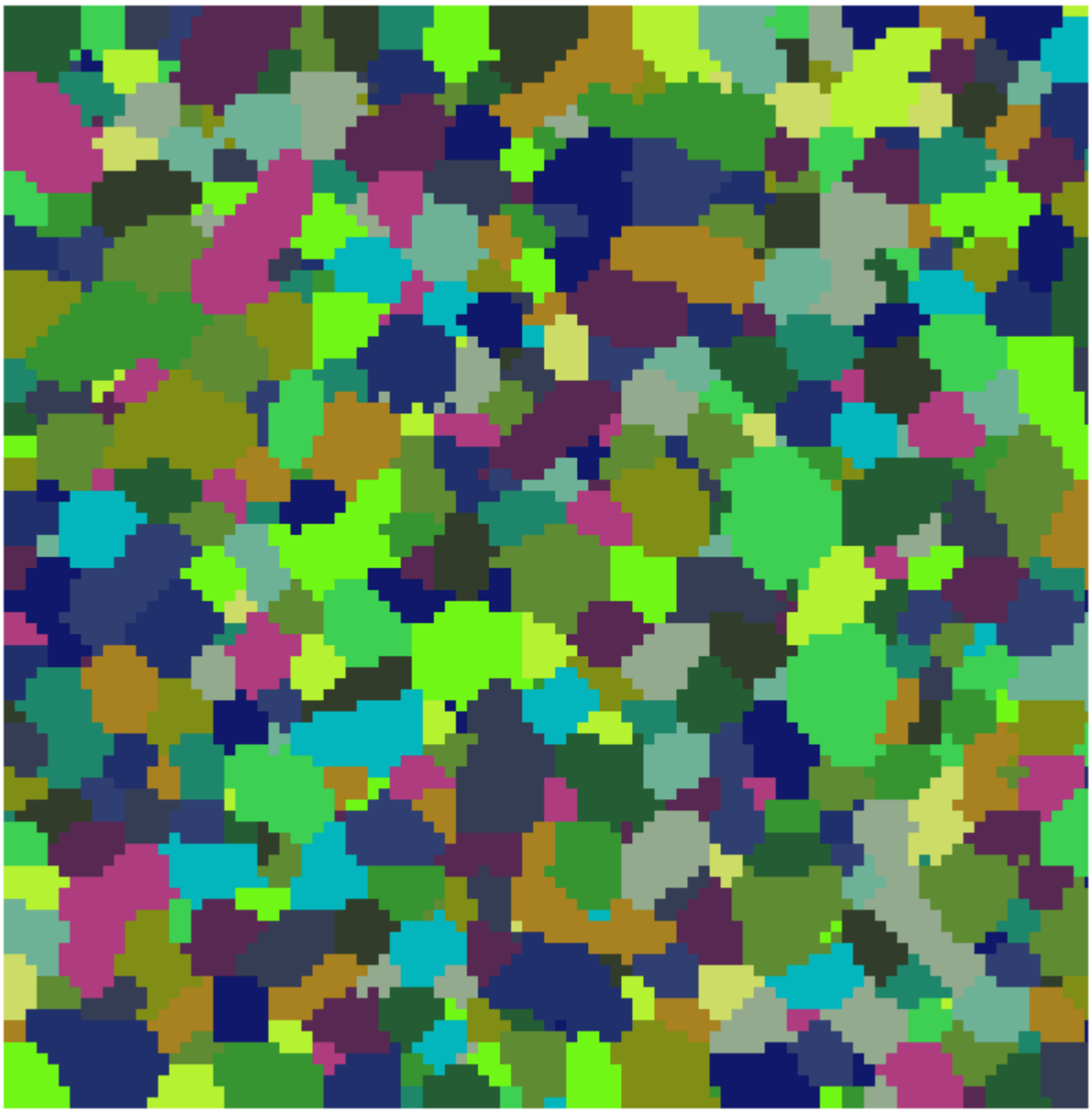
The graphical interface of the application is shown in the picture 1. It consists of buttons and fields enabling the user to enter data. They are located at the left side of the window. The right part of the window is occupied by a two canvases on which the results of the simulation will be presented.



Picture 1. Application GUI

2.2. MC grain growth

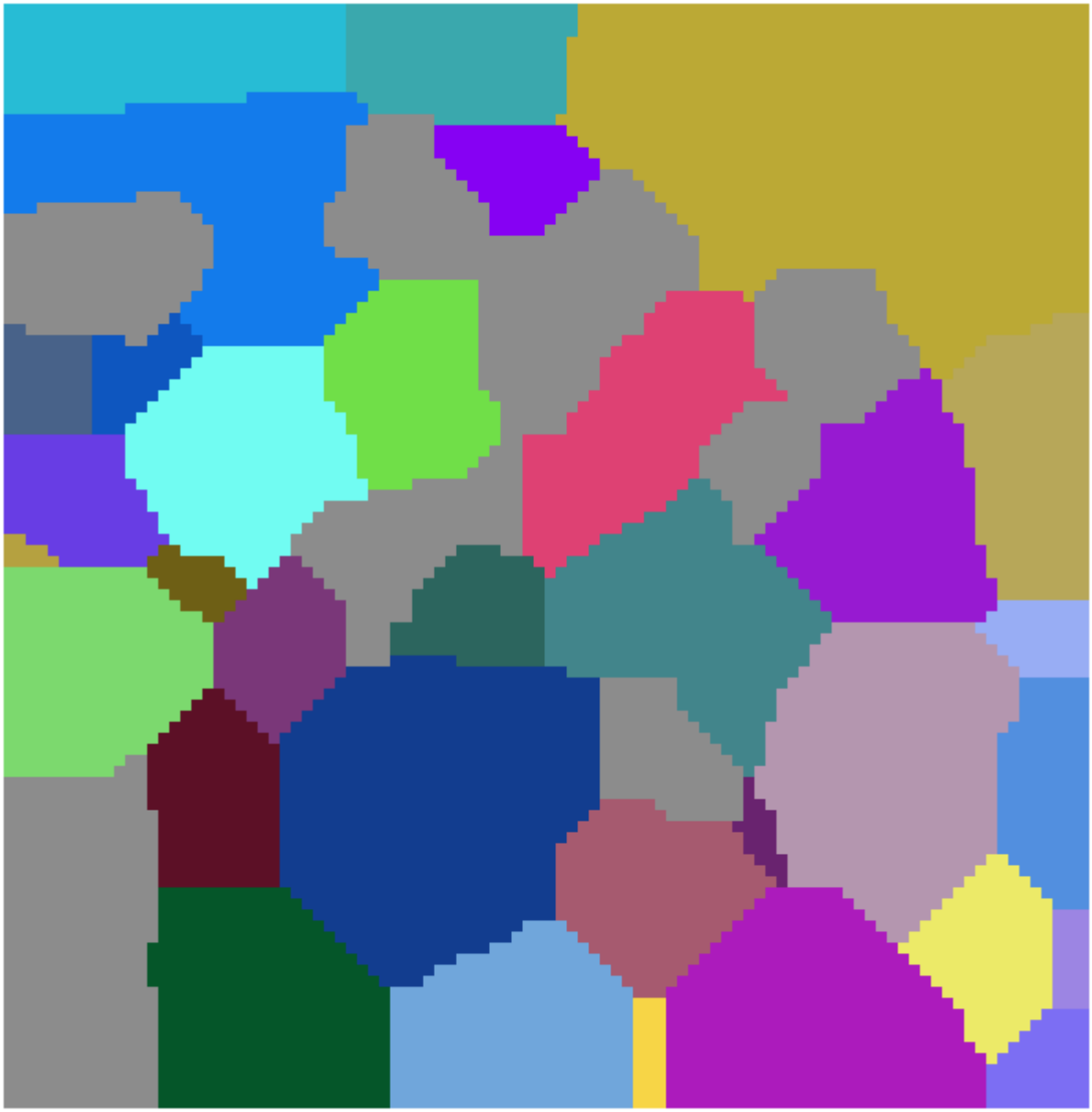
First app feature is Monte Carlo grain growth which is shown in the picture 2. To start the Monte Carlo grain growth, the user must type number of Monte Carlo iterations in **MC iterations** text box, specify grain boundary energy in **GB Energy Multiplier** text box and press the **SMC** button. The algorithm will calculate Kroneckers delta for the current grain. The Kroneckers delta is a number of grains localised in neighbourhood which have different id than the current grain. Then the programme choose randomly new id for the current grain from the set of neighbours id's and again calculate Kroneckers delta. If second Kroneckers delta is smaller than the first one, the current grain will change its id. After than the programme calculates energy for current grain by multiplying Kroneckers delta value by GB Energy Multiplier.



Picture 2. Monte Carlo Grain growth

2.3. Dual phase microstructure: CA to MC, MC to CA

Second app feature is calculating dual phase microstructure from CA to MC and from MC to CA which is shown in the picture 3. The program allows alternating simulations with the use of Cellular Automata or Monte Carlo method.

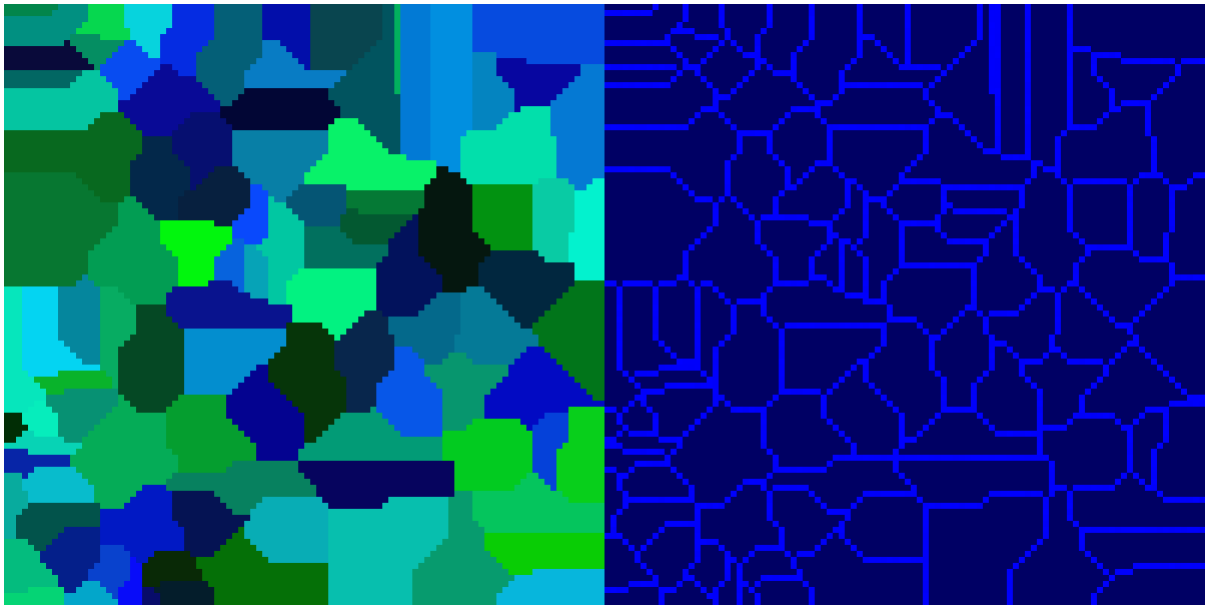


Picture 3. Dual phase microstructure with Monte Carlo grain growth

2.4. Energy distribution

Third app feature is energy distribution is shown in the picture 4. The necessary input includes: **Energy Inside, Energy Outside** and **Energy Deviation** text boxes. Thanks to this user can set the unit energy inside or/and at the boundaries of grains and add the unit energy percentage noise. Using **Set Energy** and **Show Energy** buttons user can start whole process and then show the result of it. The process of energy distribution can be conducted in two different ways. User can define it by checking **Is Homogenous** check box. First variant is heterogenous distribution.

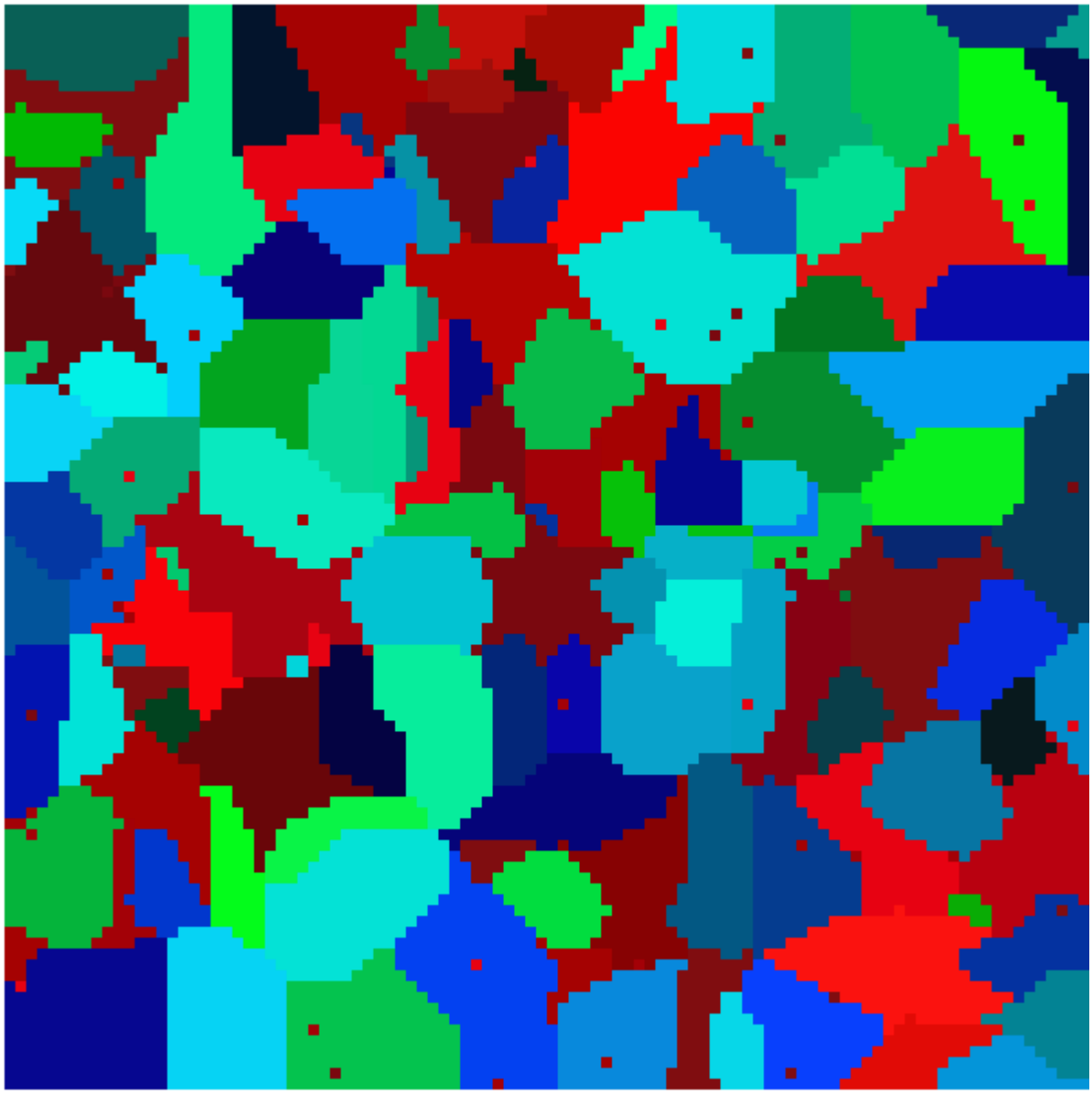
Provided this option was chosen, the inside and outside energy is inputted. Second variant is homogenous distribution. Provided this option was chosen, the only unit energy to be inputted is the inside one. After executing the energy distribution calculations a visualization window appears as it can be seen in the Picture 4. The higher energy comes in brighter colour.



Picture 4. Energy distribution

2.5. Monte Carlo static recrystallization nucleation and growth algorithm

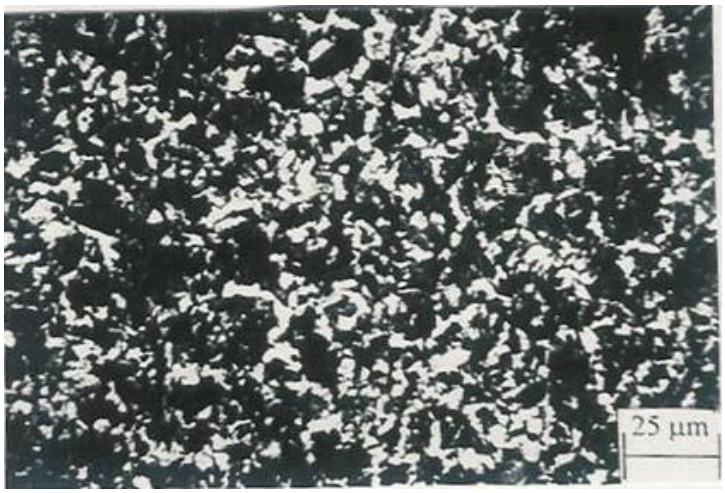
To start the algorithm the user need to specify grains amount in **Grains amount** text box and then press the **Add Recrystallized** button to add recrystalized grains to the simulation. New grains can be placed randomly or on the grain boundaries. To specify this, the user use **On grain Bounds** check box. During the simulation number of recrystallized grains can also change per iteration. User can add constant number of grains or use incrementation formula. To use one of those the user needs to simply specify amount of grains to add every iteration in **Constant** or **Inc** text box. When everything is set properly the user need to specify number of iteration in **MC iterations** text box, check **Nuc** check box and press the **SMC** button. The result of the algorithm is shown in the picture 5. The algorithm is similar to standard Monte Carlo Grains Growth algorithm, but in this case only standard grains can change their state to recrystallized one. As you can notice the recrystallized grains are coloured in shades of red.



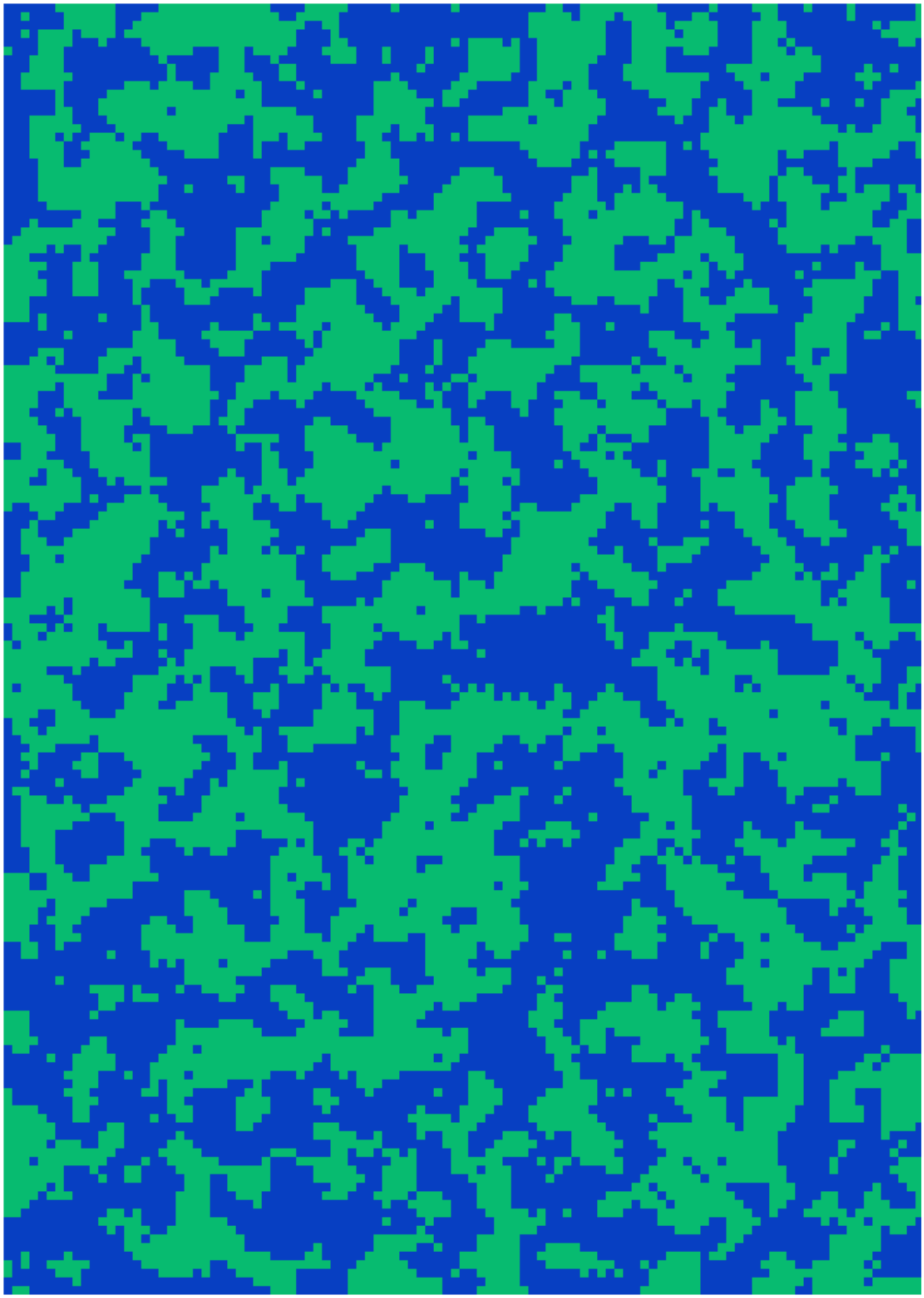
Picture 5. Result of SRX algorithm

3. Real microstructures comparison

The picture 6 shows microstructure of dual phase steel with dark islands of martensite in white matrix of ferrite [1]. The picture 7 shows the reconstruction of this microstructure achieved with the app.

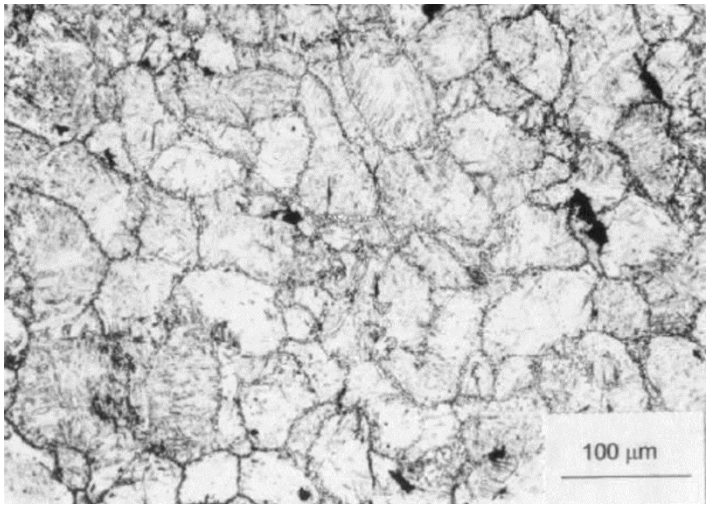


Picture 6. Microstructure of dual phase steel with dark islands of martensite in white matrix of ferrite

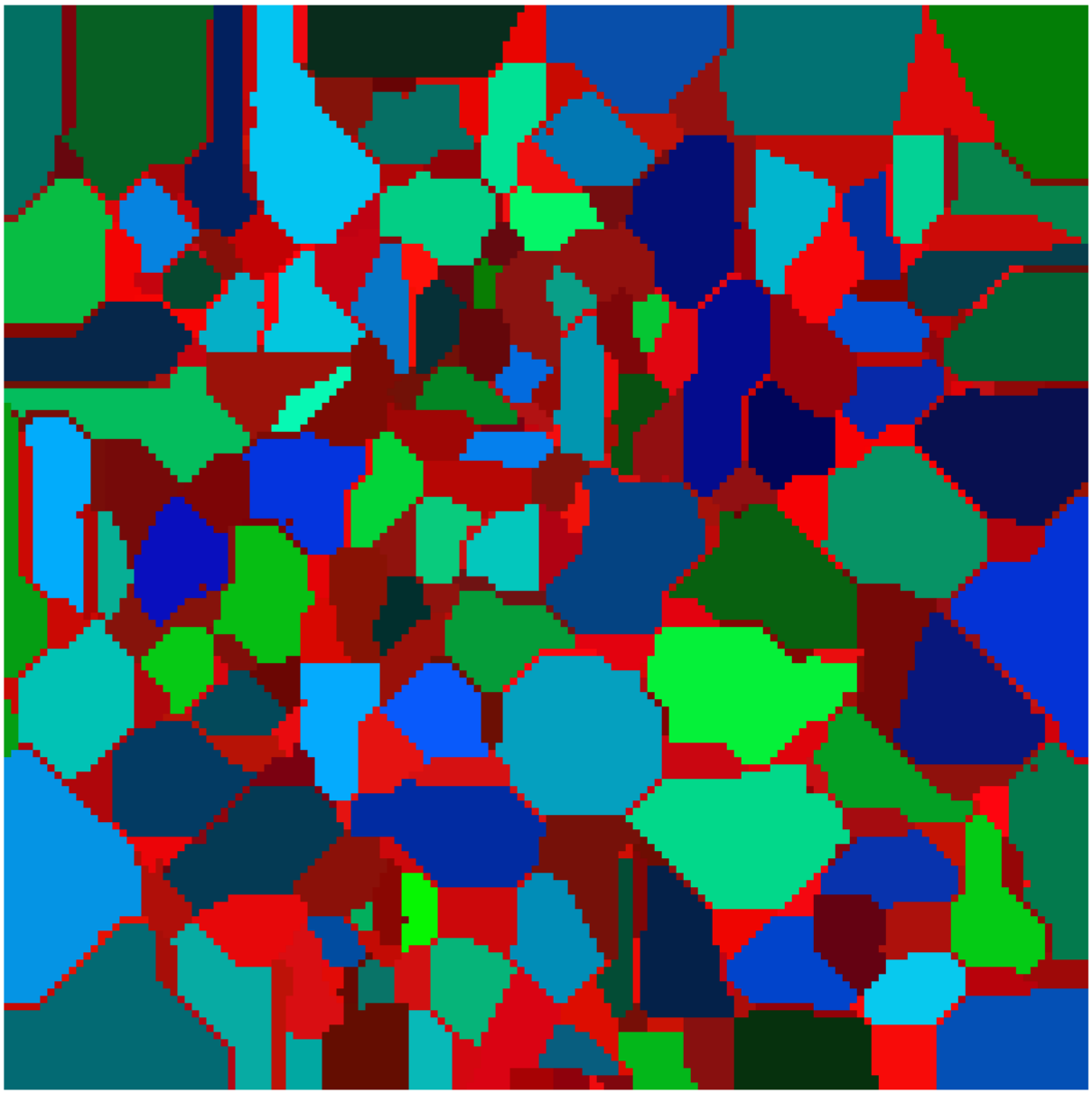


Picture 7. App generated microstructure

The picture 8 shows microstructure of recrystallized fraction in austenite of steel used reheating [2]. The picture 9 shows the reconstruction of this microstructure achieved with the app.



Picture 8. Microstructure of recrystallized fraction in austenite of steel [2]



Picture 9. App generated microstructure

4. Conclusions and summary

In some cases microstructures generated by the app are similar to the real microstructures. Given that this application can be used for simulating simple microstructures with good accuracy. Unfortunately, with the more complex microstructure, the application effectiveness drops. It is worth to mention that the Monte Carlo algorithm is more accurate in achieving complicated microstructures than the Cellular Automata algorithm. However the Monte Carlo algorithm turns out to be much slower.

5. Bibliography

[1] Microstructure of dual phase steel with dark islands of martensite in white matrix of ferrite https://www.jvejournals.com/article/20404

[2] Microstructure of recrystallized fraction in austenite of steel used reheating https://www.researchgate.net/figure/Microstructure-of-recrystallized-fraction-in-austenite-of-steel-used-Reheating_fig1_286286336