WIMiIP Studia niest. 2st 3 semestr
MultiScale Modelling – Final Report
Patryk Bogusz

Table of context:

1. Overview of the project
2. GUI Showcase
3. Results
4. Compare to real microstructure
5. Conclusion

1. Overview of the project:

Due to other deadlines and priorities the project scope was limited to up to class 3. All of the features required to fully pass each class have been satisfied. The whole project progress was monitored and stored at personal public github, url:

https://github.com/pat049b/Cellura-Automata

For each class, there is separate release made for better management of project progress, to find details please see the tab Realeses on github project.

Technology used:

For the purpose of quick development, the programming language chosen to deliver tasks is Python 3.6. To deliver GUI, the webserver Flask has been utilize, as a result the GUI is displayed in the browser. Moreover using webserver allows to deploy this application to public and let the real user use it for genereting microstructures.

Requirements to run the project locally:

- Python 3.*
- Web Browser
- **Following python packages:**
  - colour
  - flask
  - flask-expects-json
  - imageio
  - numpy
  - pillow
  - ptable
  - pytest
  - requests

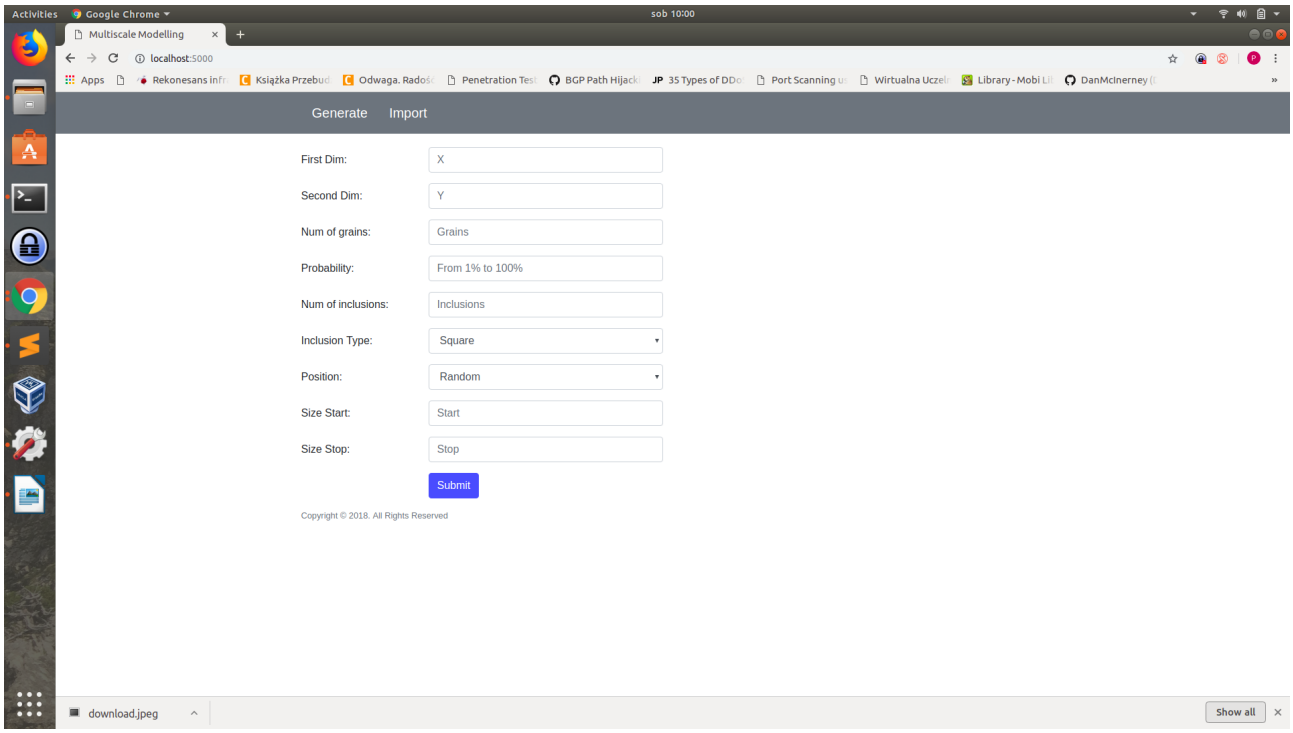
For local running there is no OS requirement because Python is multiplatform scripting language. There is also no need for Web Browser, the application could be operated by any HTTP client, saying that it can be used from the script level.

The description how to run and full documentation is avaiable at gituhub. The features delivered on each class are described with each release at github

2. GUI Showcase:

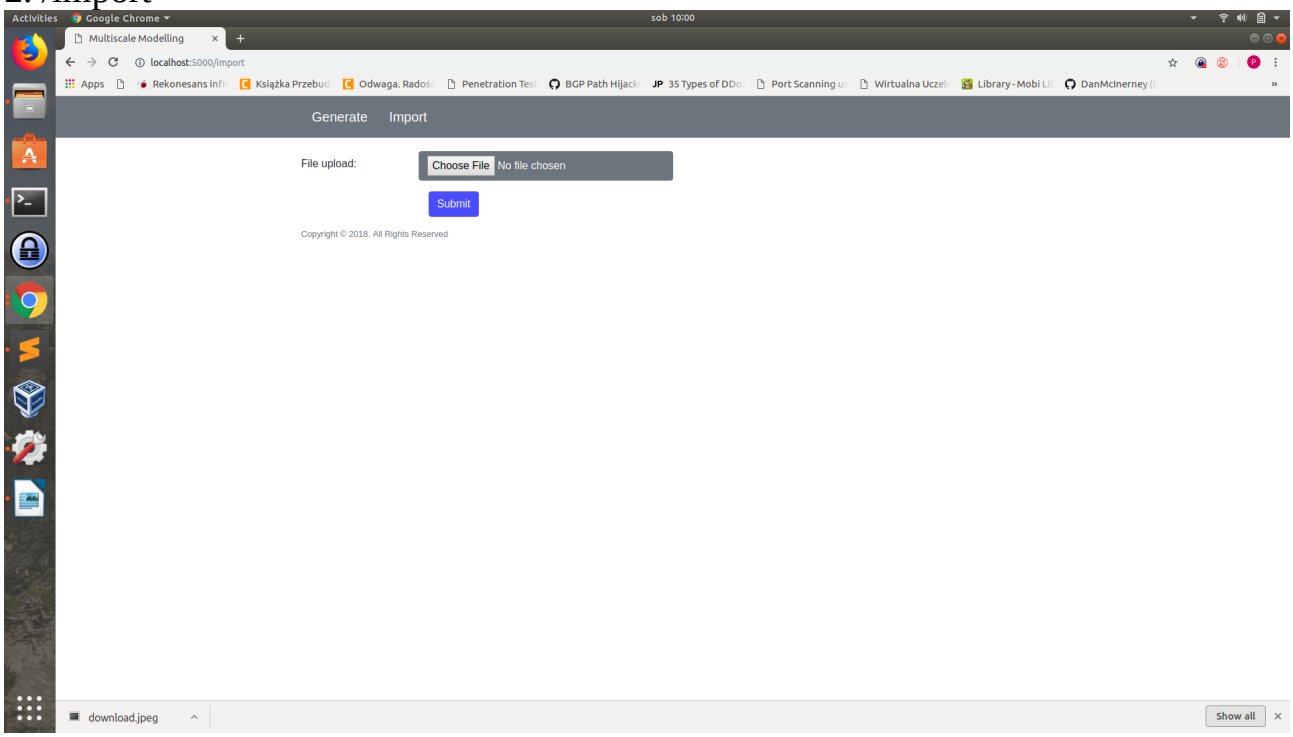
Please find bellow the showcase of the GUI. Please be informed that the application not only can be used through GUI but can be operated via any HTTP client.

1. /



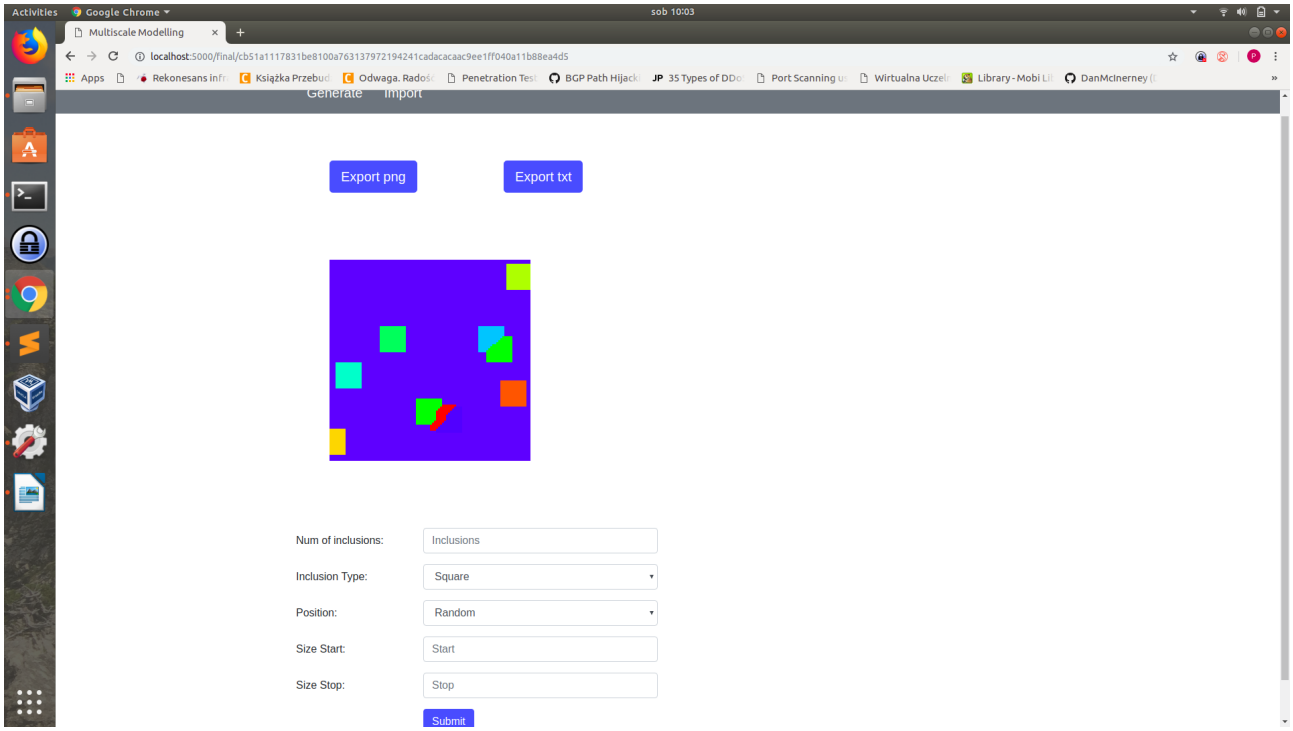
Img 1. Main Menu

2. /import



Img 2. Import Endpoint

3. *final/*<filename>



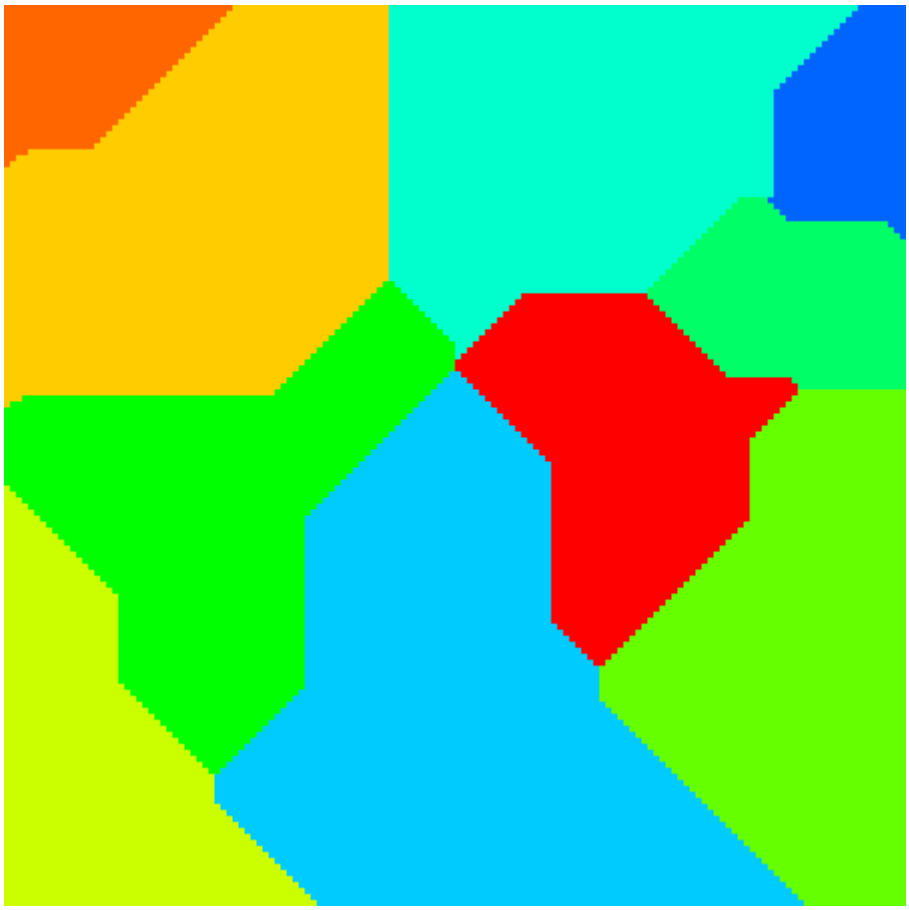
Img 3. Final Endpoint

3. Results:

It is easy to be seen that with each class the number of features grew significantly. Saying that there is huge difference between grain growth after class first and grain growth after class 3. To best coparision could be found on github in a form of gif files: https://github.com/pat049b/Cellura-Automata

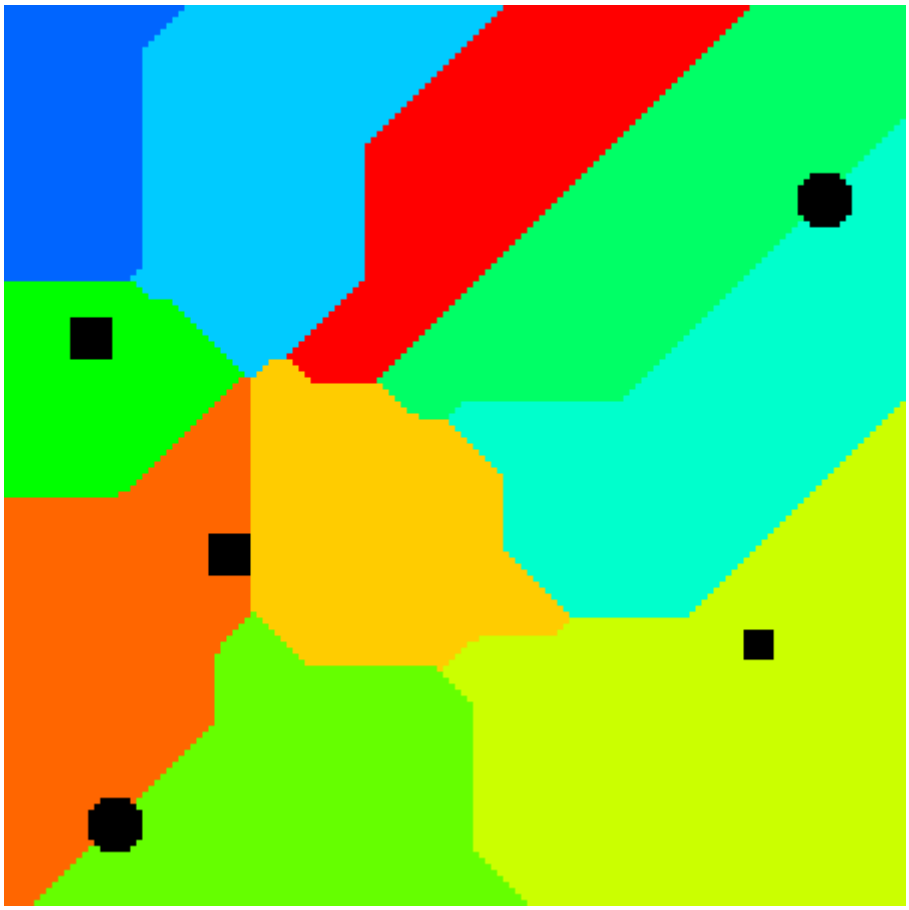
Just to show the difference in the features, please see below the screenshots showing the results of each class:

Class 1 (150x150, 10 grains):



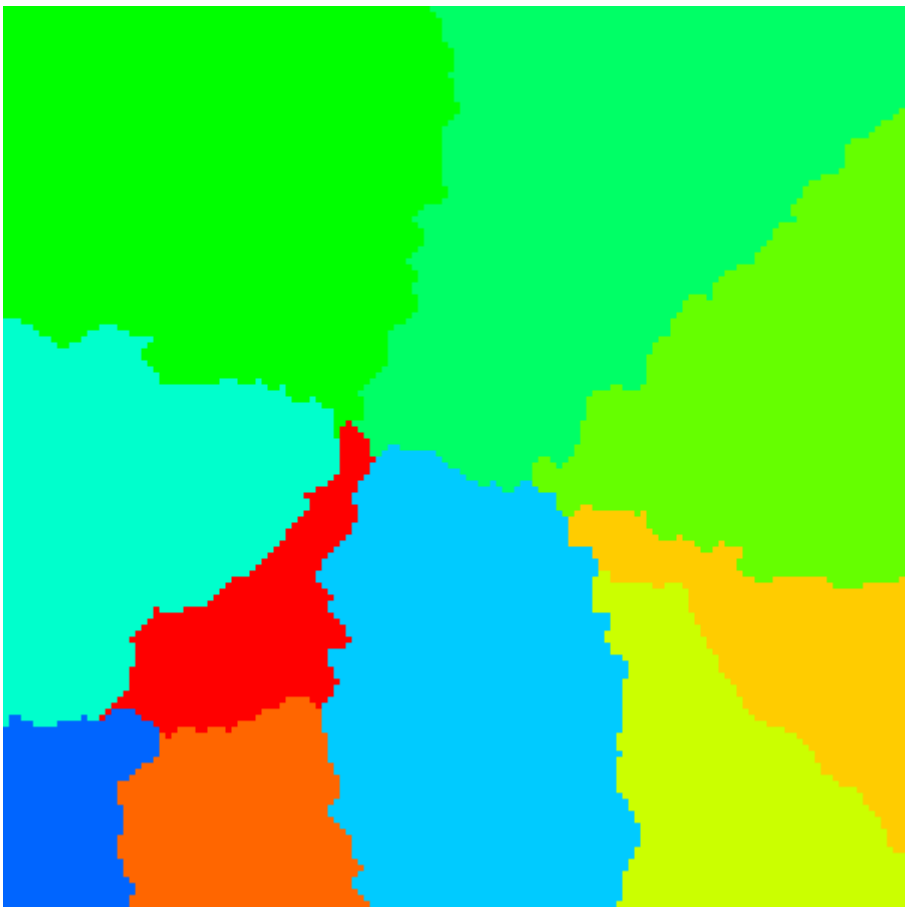
Img 4. Microstructure class 1

Class 2 (150x150, 10 grains, 3 inclusions square random R(2,3) added before the grain growth, 2 inclusions round random on boundaries R(4,5) added after grain growth):



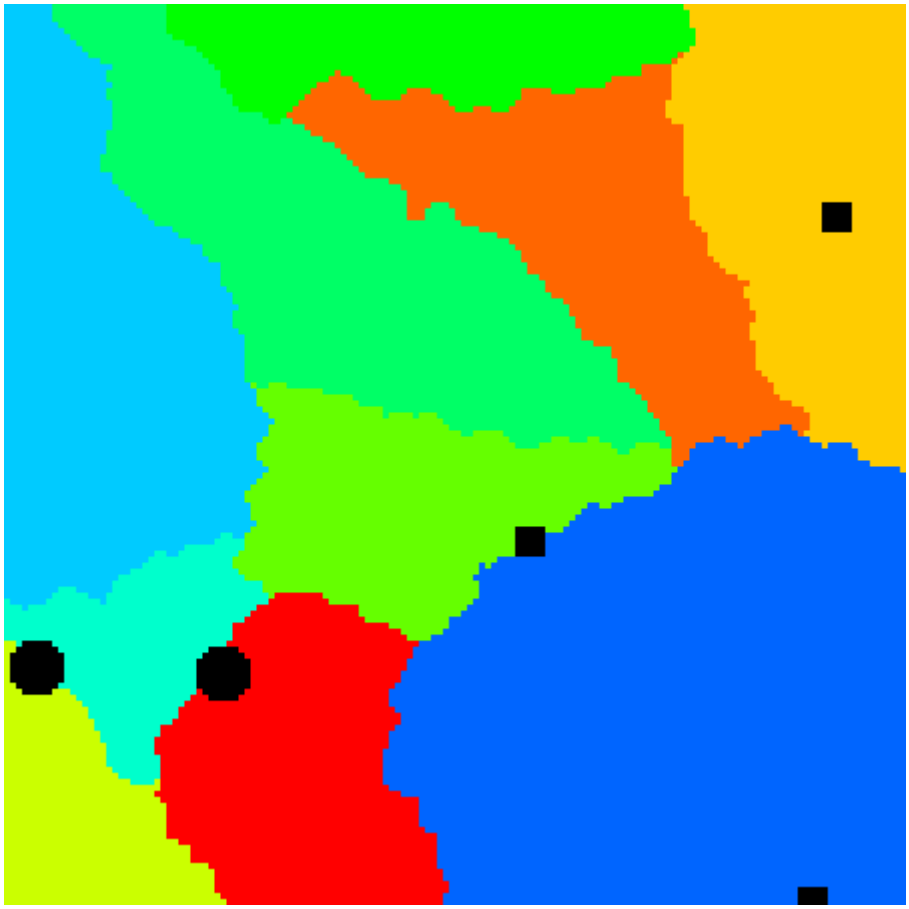
Img 5. Microstructure class 2

Class 3 (150x150, 10 grains with 20% probability):



Img 6. Microstructure class 3

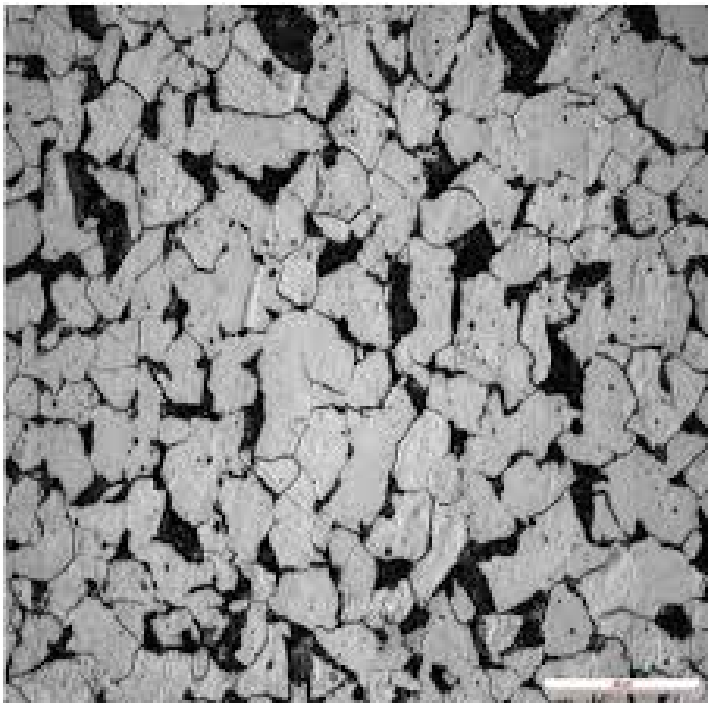
Class 3 (150x150, 10 grains, 3 inclusions square random R(2,3) added before the grain growth, 2 inclusions round random on boundaries R(4,5) added after grain growth with 20% probability):



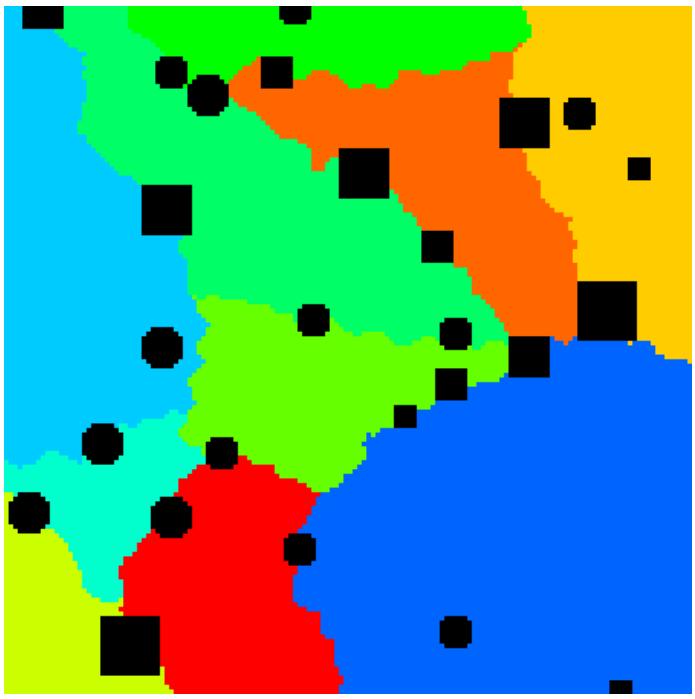
Img 7. Microstructure class 3 with inclusion

4. Compare to real microstructure:

The results from class 3 presented in previous chapter are very similar to real microstructure, please see both images below. Both of the consists of number of irregular grains and inclusions incide microstructure.



Img 8. Real Steel Microstructure
(src:https://www.researchgate.net/figure/Microstructure-of-Q235-steel-material-White-part-is-ferrite-and-black-part-is-pearlite_fig1_279457723)



Img 9. Simulated Microstructure

5. Conclusion:

The computer modelling using cellurar automata method is very useful of modelling real microstructure growth, the results showed in the previous chapter showed exactly how similar the model is to the real microstructure. The cellurar automated method might be used for modelling of other physical process.

This experience let the author also understand how using the library numpy in Python enviroment can increase the performace of software when working with big arrays. Comapring to arrays from standard Python library there have been significant speed growth when migrating to numpy arrays. Though, the Python programs are really slow compare to those written in C++ but the speed was not requirement in this project.