Curvature-driven grain growth

Master Thesis

Bastiaan van der Boor

Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

CURVATURE-DRIVEN GRAIN GROWTH

MASTER THESIS

by

Bastiaan van der Boor

in partial fulfillment of the requirements for the degree of

Master of Science
in Applied Mathematics

at the Delft University of Technology,

Student number:	4005643	
Supervisors:	dr. ir. F.J. Vermolen,	TU Delft
	dr. ir. C. Bos,	Tata Steel
Thesis committee:	Prof. dr. C. Vuik,	TU Delft
	dr. ir. F.J. Vermolen,	TU Delft
	dr. J.L.A. Dubbeldam,	TU Delft
	dr. ir. C. Bos,	Tata Steel

TU Delft Delft University of Technology

PREFACE

After nine months of work, I would like to start my thesis with thanking the people who supported me. First of all, I would like to acknowledge and thank both my supervisors for their continuous support. Dr.ir. F.J. Vermolen who helped me especially a lot in the writing process and Dr.ir. C. Bos who had the patience of teaching me how to write a C++ program.

Next I would like to thank E.M. Lazar of the University of Pensylvania who helped me overcome some difficulties in implementing his method. I would like to thank Dr.ir. S. Celotto for answering all my questions on the process steel making, when at Tata Steel. Finally I would like to thank Prof.dr.ir. C. Vuik for giving me the chance to participate in the Erasmus Mundus COSSE program.

Bastiaan van der Boor, Amsterdam 19-01-2016

CONTENTS

1

INTRODUCTION

Steel is used in numerous products and although already first used in 200 B.C. the development of new kinds of steel has never ceased to stop. More than 30 new kinds of special steel products were developed in the past year at Tata Steel alone, a result of continuous industry demand for improved steel properties. This need for new innovation is mainly driven by the automotive industry which forms the bulk of Tata Steel customers due to its strategic location in IJmuiden. That is, because car manufacturers in their race to stay ahead of competition, design and produce new car models every year. In order to produce an improved model the lightest, strongest and cheapest steel at that moment available is needed. For R&D researchers this implies a need to accelerate the development process: from drawing board to customer.

In order to be able to develop new products with certain new properties a lot of prototypes have to be produced based on experience, a time costly process. As the properties of these steels mostly depend on their microstructure, the process of developing new steels can get more sophisticated when the microstructure can be better predicted. Therefore a model has been developed by Bos et al. [1] which describes all kinds of metallurgical mechanisms occurring during the production of steel. One important mechanism is currently missing in the model: curvature-driven grain growth.

The goal of this thesis is to extent the model made by Bos et al. During the literature study [2], methods have been selected and tested on special geometries to make a well founded decision on which method describes the minimization of grain boundary energy best, as well as forms a good working combination with the model of Bos et al [1]. In the implementation a so called hybrid Cellular Automata-Lazar model is constructed. In short: the necessary data is extracted from the CA model, grain boundaries move to minimize the grain boundary energy, the changed data needs to be updated back into the CA model.

In Chapter 2 all the necessary background on the metallurgy will be presented to get familiar with all facets that play a role in the microstructure of steel. Next, the most prominent methods to model microstructures will be briefly explained. One of the boundary conditions of this research, is that the method should be implemented in the Cellular Automata method of Bos et al. Therefore each of the reviewed methods to simulate grain growth by curvature in Chapter 4 are all, at least in theory, possible to implement in the existing model. Only so called vertex methods are presented in this thesis, for other possible methods see my literature study [**?**]. One such method that has existed for some time is a method based on solving a grain boundary energy minimization problem. A relatively new method has been developed by Lazar MacPherson and Srolovitz[3, 4]

to describe grain growth in both 2D as 3D. In 2D it makes use of a theoretical relation on grain growth by curvature the so called Neumann Mullins relation. Not until recently a 3D extension of this relation did not exist making applications based on this relation lose a lot their functionality. This changed when McPherson and Srolovitz published the MacPherson-Srolovitz relation [5]. After comparison the method by Lazar, McPherson and Srolovitz has been chosen.

In Chapter 5 the implementation is discussed, some algorithms will be described. Again distinguish three cases: Extraction, movement based on Lazar, updating CA. In Chapter 6 it will be shown that the developed hybrid model works by describing some interesting cases and by giving an example of the growth of grains in 2D. Finally a summary will be given of the delivered work and some future considerations will be given.

2

CRYSTAL INTERFACES AND MICROSTRUCTURES

In the process of steel making we see a lot of changes happening on a micro-level in the steel. To better understand this process and to see the total effect of each contribution, a model can be made. We identify three major processes [6] that drive the change in microstructure and need to be taken in consideration when developing the model (where α and β denote in which phase the grains are (note: although they are of the same phase ($\alpha - \alpha$) they do not belong to the same grain):

1. Phase-transformation $\alpha - \beta$ interfaces
2. Recrystallization & Recovery $\alpha - \alpha$ interfaces
3. Grain growth by curvature $\alpha - \alpha$ interfaces

The first: phase-transformation, generally has the strongest driving force. In steel there are different phases depending on temperature and carbon concentration. Here a phase corresponds to a certain crystal structure, e.g. how the atoms are arranged. An overview of the phases with the relation to carbon concentration and temperature can be found in a phase diagram, see figure 2.1. Of special interest are the phase transformations occurring when the carbon concentration is around $0.1 - 0.7$ weight percent.

Considered are two phases: austenite and ferrite. They have the following crystal structure: Face Centered Cubic (FCC) called austenite or Body Centered Cubic (BCC) which is called ferrite. The orientation of the crystal structure can be seen in figure 2.2.

When we look at the phase-diagram it can be seen that depending on temperature and carbon concentration different crystal structures can be observed. Due to geometry of the crystal structure, a considerable amount of carbon can be dissolved. There is enough space within the structure to fit the carbon atom. While cooling down and having a low carbon concentration, the crystal will transform to a ferrite structure. In this structure less carbon can be dissolved. This seems to contradict that the crystal structure of FCC is more closely packed than BCC, since in FCC 74% of the space is filled in comparison to 68% for BCC. However the reason for this seeming discrepancy is that the size of the interstitial gaps in BCC are different, it contains a number of small gaps where the carbon atoms do not fit. Due to this difference in size, the number of gaps

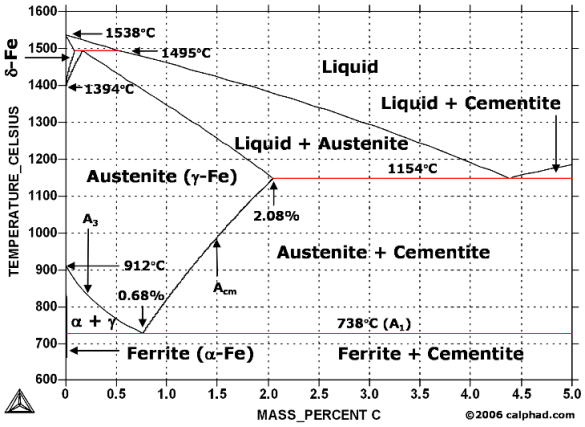


Figure 2.1: Phase diagram, source: [7]

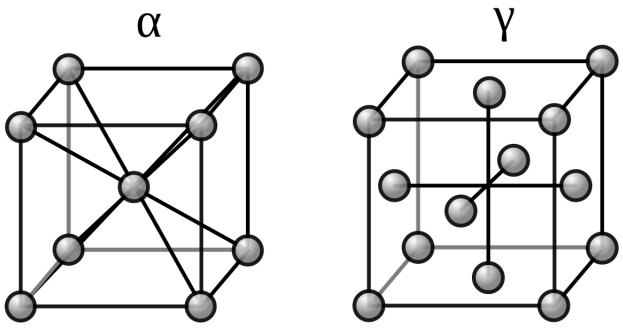
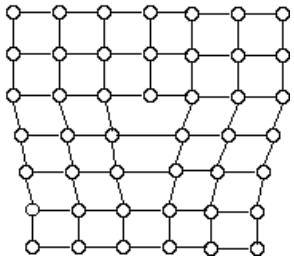


Figure 2.2: Crystal structure of Ferrite BCC α (left) and Austenite FCC γ (right), source: [8]

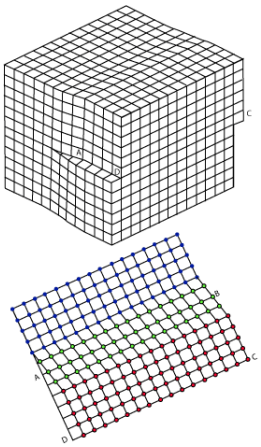
which can contain carbon, is larger in FCC than in BCC. This explains the difference in the amount of carbon that can dissolve in each phase.

Second is recrystallization and recovery, which happens simultaneously. Grain growth due to recrystallization is driven by the difference in the density of dislocation between two grains. Dislocations are irregularities in the crystal structure, two basic examples are edge- (see figure 2.3a) and screw dislocations (see figure 2.3b). Recrystallization replaces grains containing dislocations by strain-free, or dislocation-free grains. Hence, the dislocations are removed. There will be spots with a high dislocation density and where recrystallization has occurred a low density of dislocations. Recovery also depends on dislocations, in this case dislocations are removed as two dislocations with opposite direction neutralize each other. For an example see figure 2.4. In the first figure on the left as well as on the right there is an edge dislocation. The atoms on the left (green) are situated too close to each other to get the same structure as the lower grain (orange). The other way around is displayed in figure 2.4, see the middle and lower part. Two dislocations with opposite sign are present, therefore the dislocation can be removed by recovery.

Finally, grain growth by curvature is considered. This driving force of change will be the main interest of this thesis. Here, the driving force is determined by reduction in grain boundary energy. Further theoretical background explaining the factors which contribute to grain growth will be given in the remaining part of this chapter.



(a) Edge dislocation, source [9]



(b) Screw dislocation, source [10]

Figure 2.3: Two examples of dislocations

2.1. INTERFACIAL FREE ENERGY

In the past, two different sources of decrease in free energy have been proposed as driving force for grain growth. The first suggestion originally by Czochralski, who proposes a similar idea as used for crystallization [12], is the following: grains formed by recrystallization have a residual strain energy, and that upon further heating the grain with a low residual strain energy will grow at the expense of grains with a high residual strain energy. Ewing and Rosenhain [13] proposed an alternative. They suggested that the interfacial energy of the grain boundaries is the driving force for grain growth. The latter suggestion has been widely accepted as grain boundary melting stops grain growth and therefore disputing the idea of Czochralski. This can be explained as follows: at higher temperature under certain conditions the grain boundaries can melt, meaning that the atoms are no longer ordered in a crystal lattice. Note that this should not effect the growth as the residual strain energy still remains.

From Ewing and Rosenhain the interfacial free energy is needed to study grain growth. The Gibbs free energy G of a system is given by:

$$G = G_0 + A\gamma. \tag{2.1}$$

Here, G_0 is the energy of the area inside the grain, the second term is the energy contribution of the grain boundary. Further γ is the interfacial free energy per unit area and A denotes the area of the boundary.

The grain boundaries are high-energy regions that increase the free-energy of a polycrystal (solids that are composed of many grains of varying size and orientation) relative to a single crystal, polycrystalline material is never a true equilibrium structure. Therefore, due to the existence of grain boundaries a metastable equilib-

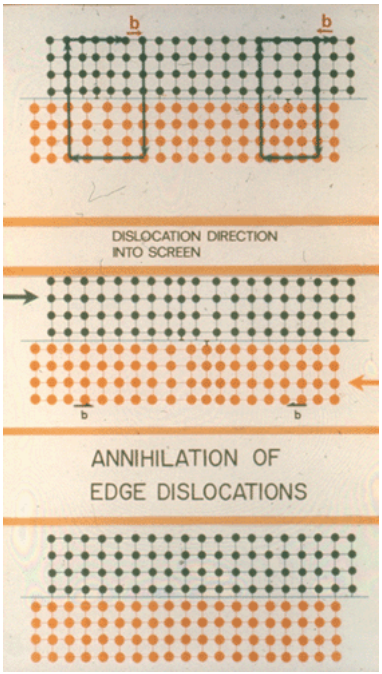


Figure 2.4: Dislocations removed by recovery, source: [11]

rium can be found. This means that a local equilibrium is found. Though, from the definition of metastability, when an activation energy is acquired a new minimum can be found. See figure 2.5 as an illustration.

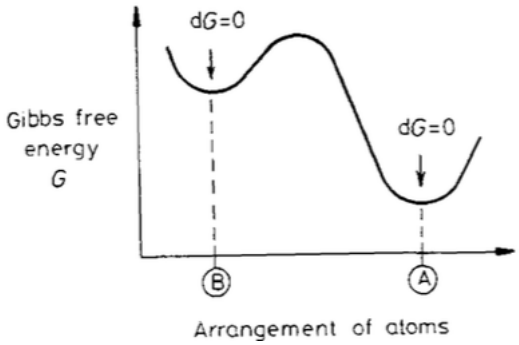


Figure 2.5: Gibbs free energy with two equilibria, source: [6]

2.2. EQUIVALENCE TO SOAP FROTH

From the observation that interfacial free energy must be minimized we can conclude that the driving force is the surface tension of the boundaries. We can defend this as follows. First it is shown by a number of authors that the shape of metal grains is equivalent to the shape of foam cells [14, 15]. Then Smith [16] showed that in a soap foam, surface tension can lead to cell growth that simulates grain growth. From the theory of surface tension which states the Gibbs free energy using the Young-Laplace equation [17]:

$$\Delta G = \gamma V(\frac{1}{R_x} + \frac{1}{R_y}). \tag{2.2}$$

The Gibbs free energy depends on γ the grain boundary energy (which will be explained later), V the molar volume and $R_{x,y}$ the radii of curvature of the grains. The grain will be similar to figure 2.6 therefore assume $R_x = R_y$. We have

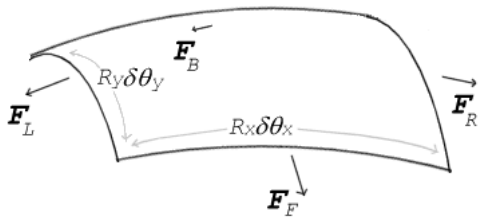


Figure 2.6: Surface tension forces acting on a small surface. $\delta\theta$ and $\delta\theta y$ indicate the amount of bend. Balancing the tension forces with pressure leads to the Young–Laplace equation, source: [18]

$$\Delta G = \gamma V(\frac{2}{R_x}). \tag{2.3}$$

It will be assumed that the velocity for grain growth will be proportional to the Gibbs free energy. We arrive at,

$$v = \alpha M \frac{2\gamma}{R_x}, \tag{2.4}$$

where α is a proportionality constant and the M the grain boundary mobility [6].

The condition for the grains to be in equilibrium can be obtained either by considering the total grain boundary energy associated with a particular configuration, or more simply by considering the forces that each boundary exerts on the junction (Chapter 3). Dunn, Chalmers and their co-workers [19, 20] made use of the effect of grain boundary melting on grain growth. They made some interesting observations which strengthens the choice for vertex-based principles. They found when melting a confined region near the grain edges, grain growth will stop. This did not necessarily happen when melting takes place near the center of a grain face. Hence, the effect from curvature near the edges are more pronounced than in the center of grain faces. Therefore, let's now only consider a vertex (intersection of three grains in 2-D).

In the case of a triple point, as the boundary tensions must be in balance, the following must hold for equilibrium:

$$\frac{\gamma_{23}}{\sin\theta_1} = \frac{\gamma_{13}}{\sin\theta_2} = \frac{\gamma_{12}}{\sin\theta_3}. \tag{2.5}$$

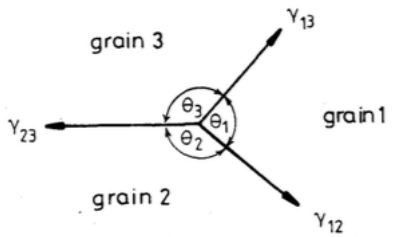


Figure 2.7: The balance of grain boundary tensions for a grain boundary triple point in metastable equilibrium, source: [6]

If grain boundary energies are assumed to be equal (this assumption will be made clear in the next paragraph). It follows that $\theta_1 = \theta_2 = \theta_3$ and therefore the angle for which the equilibrium holds, $\theta = 120°$ for a triple point. Similarly it can be shown that $\theta = 109.28°$ at a corner where four grains meet in three dimensions.

When these angular conditions are satisfied a complete meta-stable equilibrium is not yet reached. Also the surface tension must balance over all the boundary faces between the junctions. The only way the boundary tension forces can balance in three dimensions is if the boundary is:

1. Planar ($r = \infty$) or

2. Curved with equal radii in opposite directions.

Although it is theoretically possible to construct a 3-D polycrystal in which boundary tension forces are in balance, in practice this is not possible. There will always be boundaries with a net curvature in one direction. Some examples of grains with boundary tension forces are given in figure 2.8.

We see that if the number of boundaries is six and if the angles all equal 120°, the structure is metastable. Because both requisites of metastability are fulfilled: angles and also the edges connecting the vertices are straight lines. If the number of grain boundaries belonging to one particular grain is smaller than six, the edges must be concave such that the grain will shrink. If the number of grain boundaries is larger the grain will grow because its edges are convex. This will result in a decrease of the total number of grains and thereby increasing the mean grain size. Hence, the total grain boundary energy is reduced. This phenomenon is known as grain growth or grain coarsening.

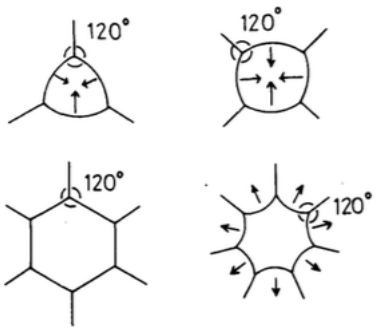


Figure 2.8: Grain boundary configurations all angles are 120°, source: [6]

2.3. GRAIN BOUNDARY ENERGY

Next the grain boundary energy will be explained. This energy comes from different orientations of a grain around the lattice. By considering two relatively simple orientations the concept will be explained. Two examples are: pure tilt boundaries and pure twist boundaries, see figure 2.9. These lattices of any two grains can be made to coincide by rotating one of them through a suitable angle about a single axis.

To consider the grain boundary energy first the assumption has to be made that the energy is proportional to the dislocations in the boundary. We have,

$$\gamma \propto D_{dis}$$

.

Here D_{dis} is the dislocation spacing. The dislocation spacing is calculated with use of the Burgers vector b and the misorientation angle θ (see for an illustration figure 2.10):

$$D_{dis} = \frac{b}{\sin\theta} \approx \frac{b}{\theta}$$

(a) Tilt

(b) Twist

Figure 2.9: Two examples of different orientations, source: [6]

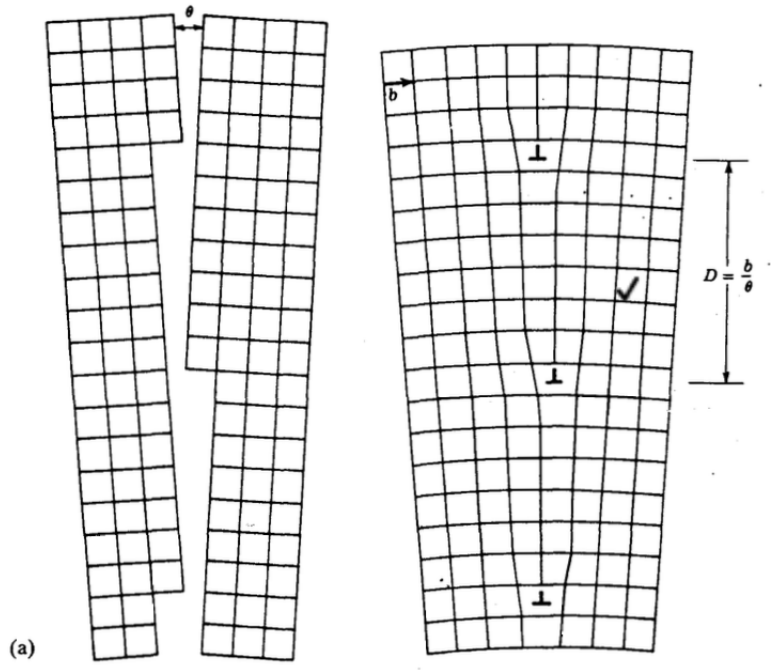


Figure 2.10: An example of a low-angle tilt boundary. Where b the Burgers vector and θ the misorientation angle, source: [6]

.

First consider a low-angle of misorientation, θ between the two grains. The dislocation spacing is very large and the grain boundary energy γ is approximately proportional to the density of dislocations in the boundary, $1/D_{dis}$. As θ increases, the strain fields of the dislocations progressively cancel out, therefore γ increases at a decreasing rate. In general when $\theta = 10 - 15°$, the dislocation spacing is so small such that the dislocation cores overlap and it is then impossible to physically identify the individual dislocations. Hence, the grain boundary energy is almost independent of the misorientation! As seen in figure 2.11.

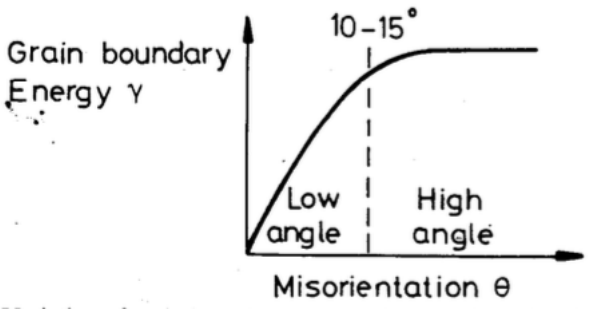


Figure 2.11: Grain boundary energy depending on the angle of misorientation, source: [6]

2.4. GOVERNING EQUATIONS

To develop an understanding of curvature driven grain growth a more mathematical approach is needed. To obtain the governing equations used in the modelling, the concept of curvature is needed. As seen in equation 2.4 the propagation of the interface depends on the radii of curvature. For any simple closed curve, curvature κ can be calculated for any given point by placing a circle which runs through the point and is tangent to points nearby (see figure 2.12). Now the governing equations of the curvature driven grain growth will be presented. In this report only the curvature flow of planar curves and two dimensional cell structures is considered.

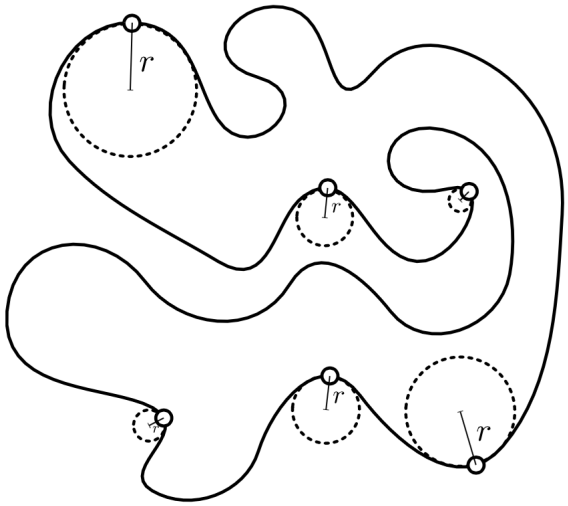


Figure 2.12: A simple, closed curve, with points highlighted, and tangent circles and radii drawn, source: [21]

A curve in the plane can be defined as a continuous mapping $\alpha : I = [a,b] \to \mathbb{R}^2$. The curve is closed if $\alpha(a) = \alpha(b)$, a curve is simple if for all $\hat{s}, u \in (a,b)$, $\alpha(t) \neq \alpha(u)$ (no intersection possible). A curve is regular if $\alpha'(\hat{s}) \neq 0 \ \forall \hat{s} \in I$. The arc-length s of a curve α mapped from the interval $[a,b]$ is defined as $s(\alpha) = \int_a^b |\alpha'(\hat{s})| d\hat{s}$. An arc-length parametrization is chosen such that the arc-length s is always exactly $b-a$.

Let an arc-length parametrized curve α be simple, closed, at least twice differentiable and $|\alpha'| = 1$ (in higher dimensions $||\nabla\alpha|| = 1$). Curvature κ can be defined as follows: Let $\mathbf{T}(s) = \frac{\alpha'(s)}{||\alpha'(s)||}$ denote the unit tangent to α at a point $\alpha(s)$, pointing in a direction in which the curve is traversed. Let $\mathbf{N}(s)$ be the unit vector normal

pointing outwards from α. Let the curve be traversed counterclockwise, then κ is defined as:

$$\kappa = \begin{cases} -||\mathbf{T}'(s)|| = -||\nabla \frac{\nabla\alpha}{||\nabla\alpha||}|| = -||\nabla^2\alpha|| & \text{if the unit tangent vector is turning clockwise} \\ ||\mathbf{T}'(s)|| = ||\nabla \frac{\nabla\alpha}{||\nabla\alpha||}|| = ||\nabla^2\alpha|| & \text{if the unit tangent vector is turning counterclockwise} \end{cases}$$

Let $\alpha(\bullet, 0) : S^1 \to \mathbb{R}^2$ be an embedded, closed planar curve that is at least twice differentiable. Then define $\alpha : S^1 \times [0, T] \to \mathbb{R}^2$, where S space and require it to satisfy:

$$\frac{\partial\alpha}{\partial t} = C\kappa\mathbf{N}, \tag{2.6}$$

as parameter t moves through $[0, T]$ the curve evolves through time. C is a constant which depends on physical properties of the system. An interesting result from Hamilton, Gage, and Grayson [22, 23] which can be observed in figure 2.13 is that every curve will finally evolve to a circle as it shrinks.

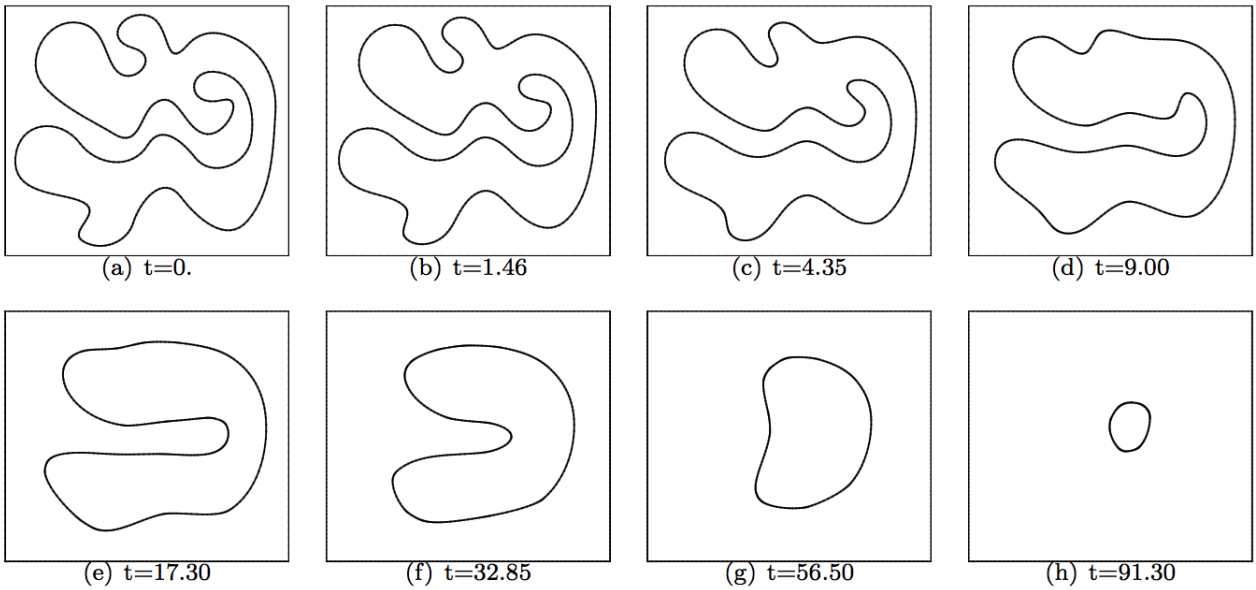


Figure 2.13: A smooth curve embedded in the plane evolves via curvature flow. The area bounded by the curve decreases at a constant rate, and the curve becomes progressively more circular; the curve eventually disappears in finite time, source: [21]

3

DIFFERENT MICROSTRUCTURE MODELS

Different models have been developed to describe all the processes occurring in the microstructure of steel. In this chapter the most prominent ones will be explained. Due to the fact that the current model is a Cellular Automata based model, the extension will have its basis formed by Cellular Automata. Notice that the different methods to calculate curvature-driven grain growth find their fundamental ideas in the microstructure models described here.

3.1. GRAIN SIZE DISTRIBUTION OF HILLERT

One of the earliest models to describe normal grain growth and the size distribution in the material was published by Hillert [24] in his paper in 1965 "On the theory of normal and abnormal grain growth". Now it is mostly used to verify the behaviour of new methods globally. He starts with equation (2.6) (Gibbs-Thompson)

$$v = M\Delta G = M\sigma(\frac{1}{R_x} + \frac{1}{R_y}). \tag{3.1}$$

Where M is the grain boundary mobility, γ the grain boundary energy and $R_{x,y}$ the principal radii of curvature. For this analysis the interest goes to the averaged value over all grains. To get the net increase of an averaged single grain a assumption has been made:

$$\frac{dR}{dt} = \alpha M\sigma(\frac{1}{R_{cr}} - \frac{1}{R}), \tag{3.2}$$

where R_{cr} is the critical size for which the grain either grows or shrinks, which can experimentally be determined and α a dimensionless constant. A steady state condition for the relative size $u = R/R_{cr}$ can be obtained from the equation (3.2). This is called the Lifshitz-Slyozov (LS) stability condition (see for a more detailed derivation the appendix of the article of Hillert):

$$\frac{du^2}{d\tau} = \gamma(u-1) - u^2, \text{ where } \tau = \ln R_{cr}^2 \text{ and } \gamma = 2\alpha M\sigma dt/dR_{cr}^2. \tag{3.3}$$

Next the Grain Size Distribution (GSD) is derived for both 2-D and 3-D case [24] and is given by:

$$P(u) = (2e)^{\beta}\frac{\beta u}{(2-u)^{2+\beta}}\exp\left(\frac{-2\beta}{2-u}\right). \tag{3.4}$$

Where $\beta = 2$ for the 2-D case and $\beta = 3$ for the 3-D case. Also the relative size u depends on the dimension, as shown in the appendix of Hillert the mean grain size is $\bar{R} = R_{cr}$ in the 2-D case.

3.2. LEVEL SET METHOD

Together with the phase field method the level set method is used in a wide region of fields [25]. The basics of the level set method in the case of grain growth due to curvature is explained here. For a more in depth and general discussion see the work of Sethian and Osher [25].

Define a smooth (Lipschitz continious) function $\phi(x,t)$, that represents the interface as the set where $\phi(x,t)=0$. Here $x=x(x_1,..x_n)\in\mathbb{R}^n$. The level set function ϕ has the folowing properties:

- $\phi(x,t)>0\ \forall x\in\Omega$
- $\phi(x,t)<0\ \forall x\notin\bar{\Omega}$
- $\phi(x,t)=0\ \forall x\in\partial\Omega=\Gamma(t)$

Thus, the interface is to be captured for all later time, by locating the set $\Gamma(t)$ for which ϕ vanishes. The motion is described by convecting the ϕ values (levels) with the velocity field $\vec{v}$. This expressed by the following equation:

$$\frac{\partial\phi}{\partial t}+\vec{v}\nabla\phi=0 \tag{3.5}$$

Only the normal component of $\vec{v}$ is needed: $v_N=\vec{v}\frac{\nabla\phi}{|\nabla\phi|}$. This gives:

$$\frac{\partial\phi}{\partial t}+v_N|\nabla\phi|=0 \tag{3.6}$$

Setting $v_N=C\kappa$, we obtain:

$$\frac{\partial\phi}{\partial t}+C\kappa|\nabla\phi|=0 \tag{3.7}$$

Now the growth of the interface is completely defined by the last equation. For a more visual illustration on how the propogating interface is described in the Level set method see figure 3.1.

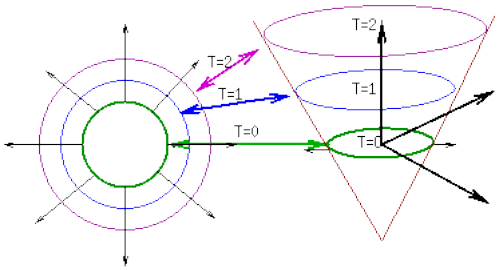


Figure 3.1: Left the original front lies in xy plane. Right the level set function, front is intersection of surface and xy plane. Here, T is the height function ϕ, source: [26]

3.3. PHASE FIELD

Here only the governing equations of the phase field method will be introduced. The main two ideas of the phase field are: a diffuse interface region (instead of a sharp one, as used in the level set method) and the minimization over the free energy. Instead of a sharp interface the phase field function $\phi(x,t)$ of thickness ϵ is introduced, which is a continuous function at the interface region. This function is for example given by:

$$\phi(x,0) = \begin{cases} 1 & \text{if x is in Grain A at time t,} \\ -1 & \text{if x is in Grain B at time t} \end{cases}$$

and at the the interface region characterized by $-1 < \phi(x,t) < 1$.

The total free energy of a system is described by the function:

$$\mathscr{F}(\phi, u) = \int_{\Omega} \left[f(\phi) + \epsilon^2 |\nabla\phi|^2 \right] d\Omega. \tag{3.8}$$

The function $f(\phi)$ gives the free energy of the system not effected by the boundary. The total free energy of the system depends heavily on the boundary and therefore the term $\epsilon^2|\nabla\phi|^2$ is added. The free-energy has to be minimized and the so called Euler-Lagrange equation is obtained. Thus minimize:

$$\frac{\delta\mathscr{F}}{\delta\phi} = 0. \tag{3.9}$$

A one dimensional example is given:

$$\frac{\partial f(\phi)}{\partial\phi} - 2\epsilon^2 \frac{d^2\phi}{dx^2} = 0. \tag{3.10}$$

The phase field method applied on the grain growth by curvature problem is found here [27].

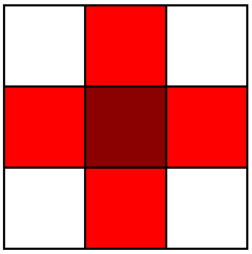
3.4. CELLULAR AUTOMATA

Already in the 1940s at the currently called Los Alamos national laboratory the first hand was laid on what is called a Cellular Automata model. Here Stanislaw Ulam studied crystal growth. He represented the crystal surfaces as nodes in a lattice network and using one of the computers built for the Manhattan Project, he observed the changes to the surfaces as the nodes obeyed the rules that he imposed on them. He suggested a similar approach on a problem his colleague von Neumann was working on. Neumann listened and adopted the suggestion and called it Cellular Automata. Nowadays it is used for a great variety of applications e.g. computer processes, cryptography, biology etc.

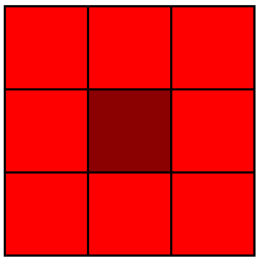
A SHORT DESCRIPTION

A short description of a general CA model is given, for a more detailed work we refer to [28]. A n-dimensional subspace is divided in a finite number of volumes, called cells. All these cells have a state assigned to it; this can be any natural number referring to the grain it belongs to. In the models described here the cell either belongs to an austenite or to a ferrite grain. The state of the cells depends on its neighbors and the state change rules defined for the considered cell. Whether a cell is neighbor cell of the cell considered depends on the neighborhood definition. Most used definitions are Neumann (left) and Moore (right) see figure 3.2. A state change rule defines the new state of a cell as a function of the states of all cells in the local neighborhood of that cell. Note that due to the flexibility of the cellular automata lots of different implementations exist!

A more mathematical description is given with the global state function Υ which is spatially partitioned into a finite number of cells. All cells i have a state ξ_i and a neighborhood η_i at discrete moments in time t_k, a state transformation function f_i^k is defined such that the new states can be computed depending on information solely from states and neighborhoods computed in previous time steps.



(a) Von Neumann neighbourhood



(b) Moore 3x3 neighbourhood

Figure 3.2: Two examples of most used neighbourhoods, source: [29]

MODEL BY BOS ET AL.

In the article of Bos et al.[1] a description is given of the model developed for the metallurgical mechanisms occurring in the annealing stage of dual-phase steels. A summary is given here, where the focus lies on which metallurgical properties are included in the model and which are not.

In the CA-model the cells form grains which have the following properties:

- The phase of the grain
- Strain energy
- Average carbon concentration
- Carbon concentration at the interface

The growth of the grains is defined by the grain-boundary velocity v, determined by:

$$v = M\Delta G. \tag{3.11}$$

Here G depends on the grain and its properties. As an example we give the interface velocity for the ferrite growth in the austenitite to ferrite phase transformation:

$$v = M_0^{\alpha\gamma} exp(-Q_g^{\alpha\gamma}/RT)\Delta G_{\alpha\gamma}(x_C^{\gamma,int}), \tag{3.12}$$

We can denote the following variables:

- $M_0^{\alpha\gamma}$, the pre-exponential factor for grain boundary mobility, the superscript denotes interface [m $J^{-1}s^{-1}$]
- $Q_g^{\alpha\gamma}$, Activation energy for the grain boundary mobility [kJmol^{-1}]
- R, the gas constant [J mol^{-1}K^{-1}]
- T, temperature in K
- $\Delta G_{\alpha\gamma}(x_C^{\gamma,int})$ [J mol^{-1}], the chemical driving force, where the variable $x_C^{\gamma,int}$ is the carbon concentration [weight %]

Recrystallization, transformation, nucleation and growth have already been included in the model, Mul [29] later changed the determination of the carbon interface concentration. Previously it was based on an exponential profile which was not accurate in the multi-grain model. The carbon concentration is now solved by using a finite difference grid on the austenite domain.

Still missing today is the growth due to surface tension or capillarity at the grain boundary. The impact of surface tension or capillarity is relatively small in the beginning of the process. When other forces stabilize the relative influence of curvature will continue to grow. This is the reason to further investigate this driving force and improve the model of Bos et al..

4

CHOOSING CURVATURE METHOD

A larger overview of the methods available to calculate grain growth by curvature can be found in the literature study. Three methods here will be discussed and compared later on to justify the decision on taking one of them to implement in the current model. These three methods are all vertex based. These methods have been preferred over counting cell methods and other derived counting cell methods for the following reasons. First, an important factor is the computation time for a counting cell method. Instead of using only a few points at the intersection of grains (vertices), for all interface cells the curvature needs to be calculated. To be able to calculate the curvature the number of cells in a neighborhood belonging to grain A or grain B are counted. When the curvature is very small the size of the neighborhood needs to be enlarged to get the necessary accuracy. This variation in size of the neighborhood will add even more complexity to the model.

4.1. DESCRIPTION OF VERTEX METHODS

The first method by Nippon, uses the grain boundary energy belonging to each particular grain to force the position of the vertices in an equilibrium state. The second method minimizes the grain boundary energy to obtain the equation of motion. The third method uses the exact solution of the total area of grain growth, with help of the Neumann-Mullins relation [30].

4.1.1. VERTEX BY NIPPON

Mathematically the most basic method is given in this thesis. Key here is the need of the correct grain boundary energy γ per grain boundary edge. Therefore only the governing equations and a short discussion of the method by Tamaki [31] will be given.

$$\mathbf{v}_{gb,i} = M_{gb,i}\gamma_i\kappa_i \tag{4.1}$$

$$\mathbf{v}_{triple,i} = M_{triple,i}\sum_{j=1}^{3}\gamma_{ij}\frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|} \tag{4.2}$$

Here $\mathbf{v}_{gb,i}$ is the velocity vector for vertices on the grain boundary (edge), so called virtual vertices and $\mathbf{v}_{triple,i}$ is the velocity vector for triple points. In figure 4.1 the method is illustrated. To obtain the curvature κ, a formula is used which calculates the radius of a circumscribed circle around three virtual vertices. The

equilibrium angles at the triple are not forced in this method as is done in the methods that will be discussed afterwards.

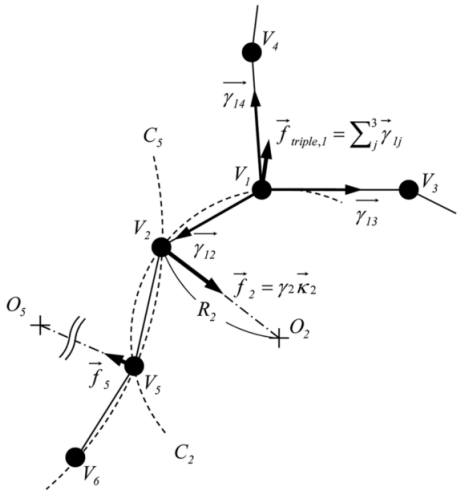


Figure 4.1: Local curvature multi-vertex model, source: [31]

4.1.2. MINIMIZATION OF GRAIN BOUNDARY ENERGY

An approach based on minimization of grain boundary energy has been proposed by Fullmann [32] in 1952. Recent developments based on Fullmann are so called vertex models. It gives the possibility to treat the influence of both the energy and the mobility of grain-boundaries. Fundamental to the vertex model is the coupling of the dissipative energy and the potential energy. The dissipative equation of motion is presented by the potential function $\mathscr{F}\{q\}$ and the Rayleigh dissipation function $\mathscr{R}\left(\{\dot{q}\},\{q\}\right)$ where $\{q\} = q_1, q_2, ...$ are a discrete set of dynamical variables [33]. Hence,

$$\frac{\partial \mathscr{F}}{\partial q_i} = -\frac{\partial \mathscr{R}}{\partial \dot{q}}. \tag{4.3}$$

This expresses the relation between the frictional force on the left and the static force on the right. To find the equations of motion for the vertices a minimization problem has to be solved to obtain the Euler-Lagrange equation. See the literature report [2] for the derivation of this relation. The following system is obtained:

$$\mathscr{D}_i \mathbf{v}_i = \mathbf{f}_i - \frac{1}{2}\sum_j^{(i)} \mathscr{D}_{ij}\mathbf{v}_j, i = 1...N \tag{4.4}$$

$$\mathscr{D}_{ij} = \frac{1}{3m_{ij}||\mathbf{r}_{ij}||}\begin{bmatrix} y_{ij}^2 & -x_{ij}y_{ij} \\ -x_{ij}y_{ij} & x_{ij}^2 \end{bmatrix}, \tag{4.5}$$

$$\mathscr{D}_i = \sum_j^{(i)} \mathscr{D}_{ij}, \tag{4.6}$$

$$\mathbf{f}_i = \frac{\partial \mathscr{F}}{\partial \mathbf{r}_i} = -\sum_j^{(i)} \sigma_{ij}\frac{\mathbf{r}_{ij}}{||\mathbf{r}_{ij}||}. \tag{4.7}$$

Where j in the sum $\sum_j^{(i)}$ is running over the vertices which are connected to vertex i and N is the total number of vertices in the system. In Kawasaki [33] the system of equations is further simplified. This will not be done as the convergence to equilibrium state is not happening because of the simplification according to D. Weygand [34].

4.1.3. VERTEX METHOD USING NEUMANN-MULLINS EQUATION

In 2009 the article "A more accurate two-dimensional grain growth algorithm" has been published by Lazar, MacPherson and Srolovitz [4]. Here the Neumann-Mullins relation [30] (an exact relation for grain growth in two dimensions using the number of vertices) is used to obtain a method describing the grain growth due to surface tension. Although a two-dimensional algorithm is in theoretical perspective use full in practice it is not. Not until 2007 an extension to three dimensions was not available. This changed when Robert D. MacPherson together with David J. Srolovitz published the article "The von Neumann relation generalized to coarsening of three-dimensional microstructures" in Nature [5]. For now only the two dimensional case is considered, but extending to three dimensions seems evident [3].

First the proof of Neumann-Mullins relation will be discussed. Then the results of [30] will be used to obtain a method for grain growth in two dimensions, as proposed in [4].

DERIVIATION OF NEUMANN-MULLINS EQUATION

In 1956 Mullins published the article called: "Two dimensional motion of idealized grain boundaries" [30]. Here he proofed a general theorem concerning the change of area enclosed by a curve in two dimensions. This curve is defined as $r(\theta,t)$, where r and θ are polar coordinates and t time. Assumed is that any point of the curve moves towards it center of curvature with a velocity v, given by $v = M\gamma\kappa$, where M is the grain boundary mobility and γ the grain boundary energy. Define the arc length s of a curve α as $s(\alpha) = \int_a^b |\alpha'(t)|dt$ and β the angle measured in a counterclockwise manner between the positive x-axis and the directed tangent. From figure 4.2 it can be seen that $\Delta r \sin\psi = -M\gamma\kappa\Delta t$. Using the relations $\kappa = \frac{\partial\beta}{\partial s}$ and $\sin\psi = r(\frac{\partial\theta}{\partial s})$ (from Green's Theorem) a new relation for $\frac{\partial r}{\partial t}$ can be obtained:

$$\frac{\partial r}{\partial t} = -M\gamma\frac{\kappa}{\sin\psi} = -M\gamma\frac{1}{r}\frac{\partial\beta}{\partial\theta}. \tag{4.8}$$

The area enclosed by a simple closed curve is given by $A = \frac{1}{2}\oint_0^{2\pi} r^2 d\theta$, where the integral is taken counterclockwise around the curve.

Using these two results and differentiating over time gives:

$$\frac{dA}{dt} = \oint \frac{\partial r}{\partial t} r d\theta = -\oint M\gamma\frac{\partial\beta}{\partial\theta}d\theta = -M\gamma\oint d\beta = -2\pi M\gamma. \tag{4.9}$$

Remark: the result also holds for M,γ being a function of β as the function is 2π-periodic.

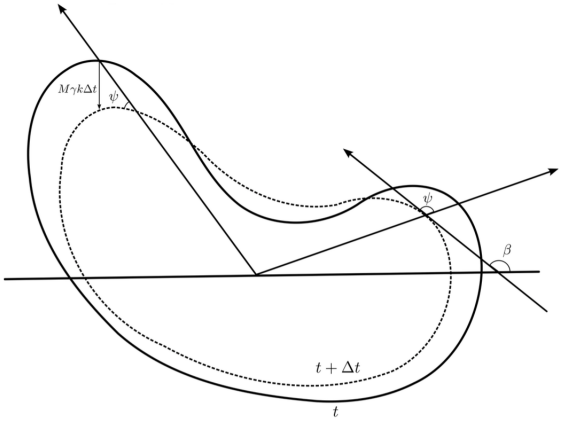


Figure 4.2: Closed curve which moves towards it center of curvature, source: [30]

From this relation it can be concluded that two curves of different shape, but enclosing equal areas have an equal growth rate!

Now a similar relation will be obtained for a two dimensional area of grain boundaries. A network of arbitrary curves dividing the plane into polygon-like grains is presented, see figure 4.3. As an illustration a grain with n sides is considered, the vertices are numbered counter clockwise, each side has the same number corresponding to the preceding vertice. Let β_{ij} be the angle between the x-axis and the tangent to i-th side and through the j-th vertex. The angle between each edge is assumed to be $\frac{2\pi}{3}$ or 120 degrees, as explained in Chapter 1. When every edge has a different grain boundary energy (anisiotropic), a different equilibrium angle can be derived. Using the relation of equation (4.9), we find the rate of change of area of the grain to be:

$$\frac{dA}{dt} = -M\gamma \oint d\beta, \tag{4.10}$$

$$= -M\gamma \left[(\beta_{11} - \beta_{12}) + (\beta_{22} - \beta_{23}) + (\beta_{33} - \beta_{34}) + \ldots + (\beta_{nn} - \beta_{n1}) \right], \tag{4.11}$$

$$= -M\gamma \left[(\beta_{22} - \beta_{12}) + (\beta_{33} - \beta_{23}) + \ldots + (\beta_{11} - \beta_{n1}) \right]. \tag{4.12}$$

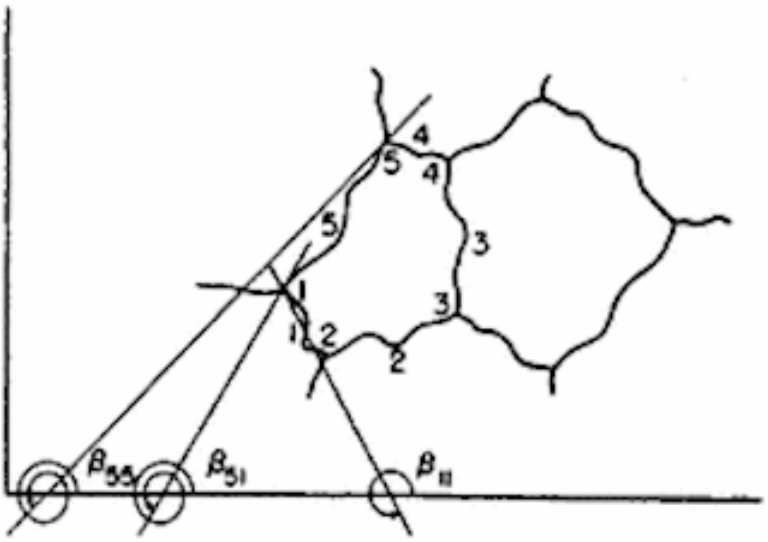


Figure 4.3: Two dimensional area of grain boundaries, source: [30]

From figure 4.4 it can be concluded that $\beta_{22} - \beta_{12} = \pi - \frac{2\pi}{3} = \frac{\pi}{3}$ which is the case for the first $n-1$ terms. From figure 4.3 $\beta_{11} - \beta_{51} = \frac{\pi}{3} - 2\pi$. Hence,

$$\frac{dA}{dt} = -M\gamma \left[(n-1)\frac{\pi}{3} - 2\pi + \frac{\pi}{3} \right] = -M\gamma \frac{\pi}{3}(6-n). \tag{4.13}$$

IMPLEMENTATION OF NEUMANN-MULLINS RELATION IN VERTEX METHOD BY LAZAR ET AL. [4]

The following idea is considered: use the knowledge of the total grain growth (Neumann-Mullins relation, which is the exact for normal grain growth in an isotropic polycrystal) of one particular grain and divide this growth over all the vertices.

For simplicity first a single isolated grain is considered see figure 4.5, hence $n = 0$. A key observation is that when changing the area locally the total area is changed with the same amount. When for example

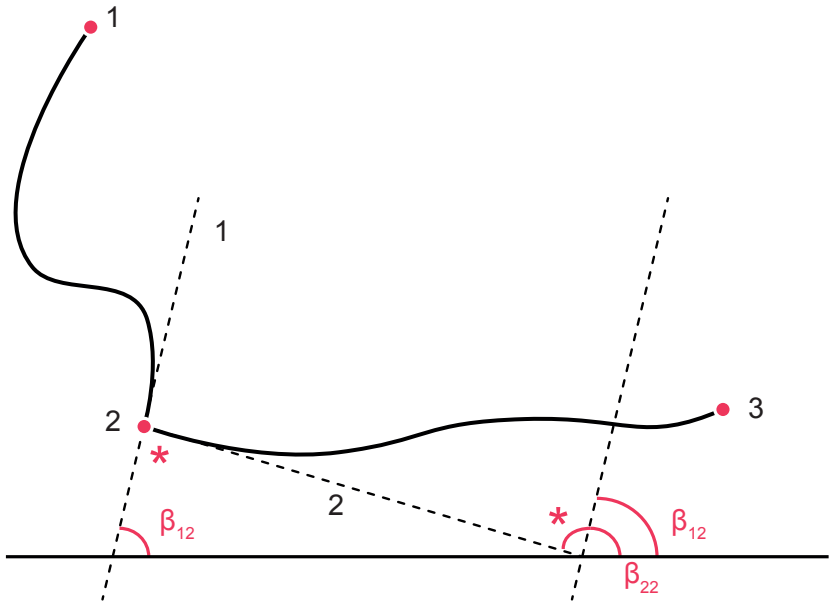


Figure 4.4: A closer look on how the β is constructed

the position of the vertex σ is changed, only the gray area is affected and the total area of the grain changes with the same amount. Therefore when using the derived relation above (4.13) and by summing over over all nodes separately:

$$\Delta A = 2\pi M\gamma(\frac{0}{6} - 1)\Delta t = -2\pi M\gamma\Delta t \tag{4.14}$$

$$= \sum_{i=1}^{n} -\alpha_i M\gamma\Delta t, \text{ where } \sum_{i=1}^{n} \alpha_i = 2\pi. \tag{4.15}$$

First the equation of motion of a virtual vertex is derived. This is a vertex which has only two neighbouring grains and therefore only has two neighbouring vertices. If only virtual vertices exist $n = 0$ and a local change in area by moving a vertex is exactly the same as the global change in area. Consider a node σ with edges $\mathbf{e_1}$ and $\mathbf{e_2}$ as shown in Figure 4.5. The exterior angle α_σ, from the cosine law $\alpha_\sigma = \cos^{-1}\left(-\frac{\mathbf{e_1}\mathbf{e_2}}{|\mathbf{e_1}||\mathbf{e_2}|}\right)$. Node σ is moved by a displacement vector $\mathbf{v}_\sigma$, which will change the area of the triangle by $\Delta A = -\alpha_i M\gamma\Delta t$ (using equation (4.14)). For numerical stability reasons node σ is moved in the direction of $\mathbf{e_1} + \mathbf{e_2}$. Hence,

$$\mathbf{v}_i = \alpha_i M\gamma\Delta t \frac{\mathbf{e_1} + \mathbf{e_2}}{|\mathbf{e_1} \times \mathbf{e_2}|}. \tag{4.16}$$

Consider the triple junction node τ (or node i in general) as shown in figure 4.6. Again the Neumann-Mullins relation is considered, now for the displacement of the triple junction node. Because there are three different grains meeting at a triple point the relation $\Delta A_j = -(\alpha_j - \frac{\pi}{3})M\gamma\Delta t$ holds, where α_{ji} is the exterior angle at the triple junction with respect to grain j. When one grain has n triples this means it also has exactly n neighbouring grains. Therefore,

$$\sum_{i=1}^{n}(\alpha_{j,i} - \frac{\pi}{3})M\gamma\Delta t = (2\pi - \frac{n\pi}{3})M\gamma\Delta t, \tag{4.17}$$

$$= -2\pi M\gamma(1 - \frac{n}{6}). \tag{4.18}$$

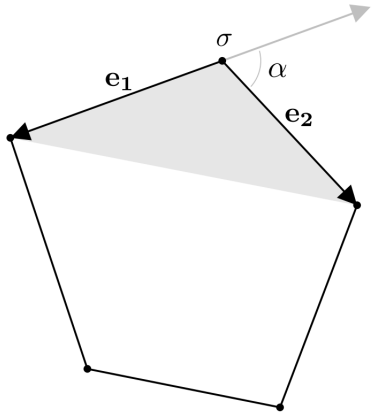


Figure 4.5: Single grain, node σ with displacement vector and edges $\mathbf{e}_1$,$\mathbf{e}_2$ connected to σ, source: [4]

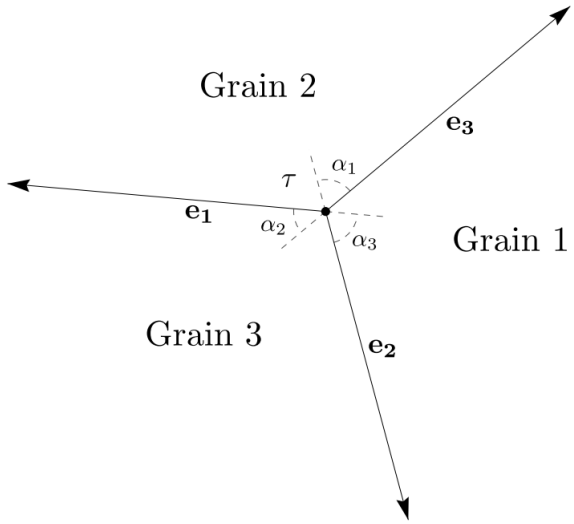


Figure 4.6: Triple defined by its vectors belonging to the edges connected to the triple and exterior angles α_{ji}, source: [4]

The α's are given by:

$$\alpha_{1,i} = \cos^{-1}\left(-\frac{\mathbf{e_{1,i}}\mathbf{e_{2,i}}}{|\mathbf{e_{1,i}}||\mathbf{e_{2,i}}|}\right), \tag{4.19}$$

$$\alpha_{2,i} = \cos^{-1}\left(-\frac{\mathbf{e_{2,i}}\mathbf{e_{3,i}}}{|\mathbf{e_{2,i}}||\mathbf{e_{3,i}}|}\right), \tag{4.20}$$

$$\alpha_{3,i} = \cos^{-1}\left(-\frac{\mathbf{e_{3,i}}\mathbf{e_{1,i}}}{|\mathbf{e_{3,i}}||\mathbf{e_{1,i}}|}\right). \tag{4.21}$$

For all three grains the Neumann-Mullins relation needs to hold. Hence, the following system of equations need to be solved:

$$\begin{bmatrix} \mathbf{e_{1,i}} - \mathbf{e_{2,i}} \\ \mathbf{e_{2,i}} - \mathbf{e_{3,i}} \\ \mathbf{e_{3,i}} - \mathbf{e_{1,i}} \end{bmatrix} \mathbf{v}_i = 2M\gamma\Delta t \begin{bmatrix} \alpha_{1,i} - \frac{\pi}{3} \\ \alpha_{2,i} - \frac{\pi}{3} \\ \alpha_{3,i} - \frac{\pi}{3} \end{bmatrix}. \tag{4.22}$$

Due to the fact that there are three equations and the degree of freedom is two. The number of equations can be reduced, obtain:

$$\mathbf{v}_i = 2M\gamma\Delta t \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{e_{1,i}} - \mathbf{e_{2,i}} \\ \mathbf{e_{3,i}} - \mathbf{e_{1,i}} \end{bmatrix}^{-1} \begin{bmatrix} \alpha_{1,i} - \frac{\pi}{3} \\ \alpha_{3,i} - \frac{\pi}{3} \end{bmatrix}. \tag{4.23}$$

4.2. COMPARING VERTEX METHODS

After discussing three possible methods, now a decision has to be made which method will be implemented in the general model. To substantiate the decision different geometries will be tested.

Distinguish two ways of testing whether the method works appropriately. First the most important one, the methods are compared with the analytic solution. The vertices are placed on a circle where the exact curvature is known; hence comparison with the exact solution is possible. The second test is comparing the found solution with an analytical implementation of the method on the selected test case. This has been done by substituting an analytical representation of the problem in the method [34].

SIMPLE POLYGON

First a polygon where the vertices are placed on a circle is constructed. Knowing the exact curvature of a circle, the displacement of a vertex using different methods can be compared with the displacement according to the Gibbs-Thompson effect: $v = -M\gamma\kappa$.

Another way is introduced by Weygand et al. [34], he makes use of equation (4.4) to find the analytic solution of the change in total area of a n-sided polygon. See appendix A to see the derivation which leads to:

$$\frac{dA'_n}{dt} = -2M\gamma n \tan(\frac{\pi}{n}). \tag{4.24}$$

Take the limit of $n \to \infty$ it can be seen that it is similar to the exact solution of a circle:

$$\lim_{n\to\infty} \frac{dA'_n}{dt} = -2\pi M\gamma, \tag{4.25}$$

which is the Neumann-Mullins relation for $n = 0$. Note: Here the dissipation and potential term is used as a starting point, which of course is already an approximation of the influence of the surface tension.

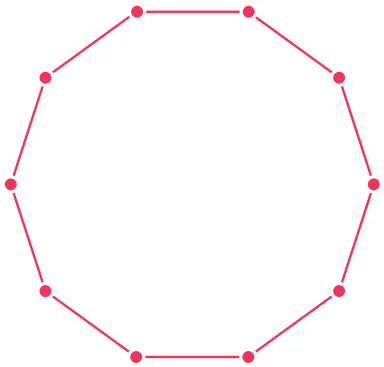


Figure 4.7: n-sided polygon

EMBEDDED POLYGON

The second test case is an extended version of the simple polygon. Whereas in the simple polygon only the effect on virtual vertices could be observed by adding more vertices, also triple points can be studied. A set of vertices with fixed locations is placed equidistant from each other on the outer circle. These vertices are connected to the closest already existing vertices. Hence, a set of triple points is constructed, see figure 4.8.

Using the Neumann-Mullins relation (equation (4.26)), an exact solution to the change in area of the polygon is available. This will be compared with the change in area of the polygon which is embedded in the polygon constructed by the newly added vertices.

$$\frac{dA}{dt} = -2\pi M\gamma(1 - \frac{1}{6}n) \tag{4.26}$$

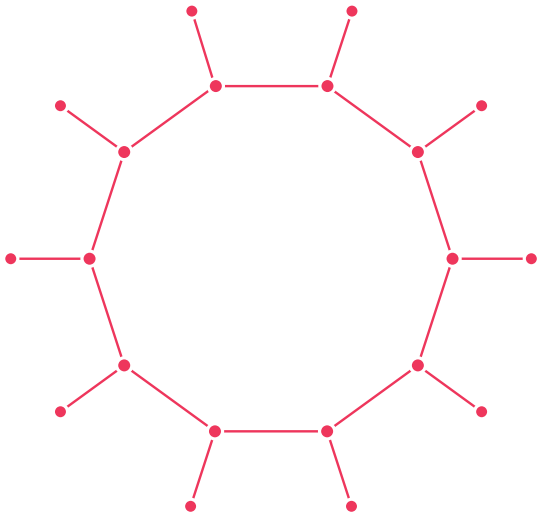


Figure 4.8: Embedded polygon, triples in the inner polygon

EMBEDDED POLYGON WITH VIRTUAL VERTICES

To see the effect of adding virtual vertices, an embedded polygon is expanded with so called virtual vertices. In figure 4.9 only one extra vertex is placed between each triple point, but this number can vary (take in account the discussion in Chapter 4.1). The effect of small perturbations of the vertices can now be observed. Another interesting result would be to see what happens to the vertices when the number of triple points is six. From Neumann-Mullins the total area of the polygon should not change in time, as a hexagon is in equilibrium state.

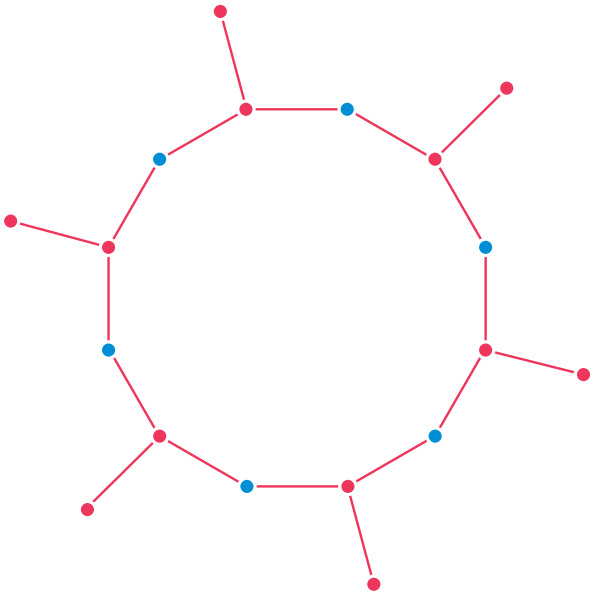


Figure 4.9: Embedded polygon with $n_{virtual} = 1$ (blue)

4.3. RESULTS

The results of the test cases introduced in Chapter 4 will be presented and discussed. The differences between the method first presented by Kawasaki and the method by Lazar is highlighted, such that a well reasoned decision can be made on which method to finally implement.

A SIMPLE POLYGON

As expected the growth of the polygon by Lazar's method is consistent with the Neumann-Mullins relation:

$$\frac{dA}{dt} = -M\gamma\frac{\pi}{3}(6-n) = -M\gamma 2\pi \tag{4.27}$$

The Method of Tamaki is simply calculating the curvature by finding the unique circle based on its two neighbouring points and the considered vertex (the essence of the method by Nippon when only virtual vertices are implemented). As seen in figure 4.10 this coincides with Kawasaki and when enough vertices are introduced it is similar to the result obtained by Lazar's method, as was expected see equation (4.25).

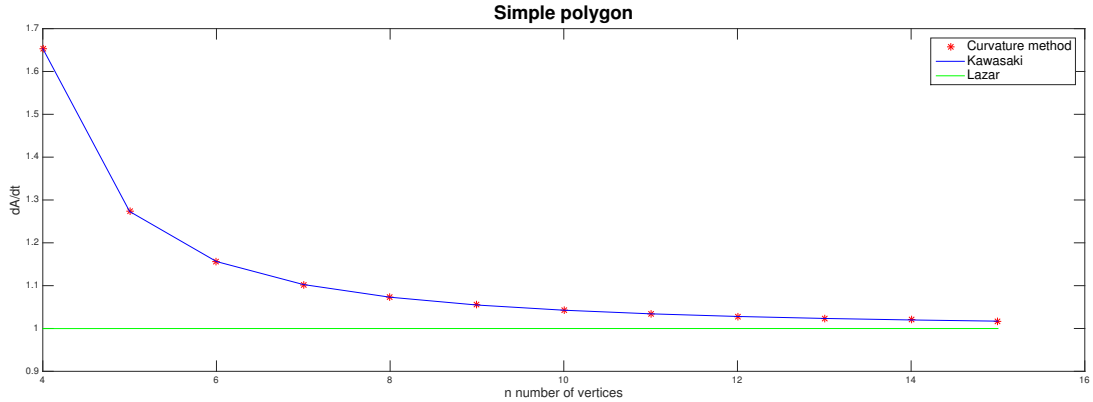


Figure 4.10: Change in area of a simple polygon, the derivative of area is multiplied by the factor $-\frac{1}{2\pi}$ and $M = 1$

EMBEDDED POLYGON

The embedded polygon introduced in Chapter 4 is a different interpretation of the embedded polygon introduced in Weygand et al. There the virtual vertices are placed on the straight line connecting the triples. When implementing this it can be seen that figure 4.11 coincides with figure 5 of Weygand et al. However, when placing the virtuals on the circle as is done in figure 4.12 for Lazar's method, the method gives a large error when comparing to Neumann-Mullins relation. Observed can be that this is not the case for Lazar's method. When implementing the method of Nippon on the polygons and comparing with the Neumann-Mullins relation, no sensible results could be found.

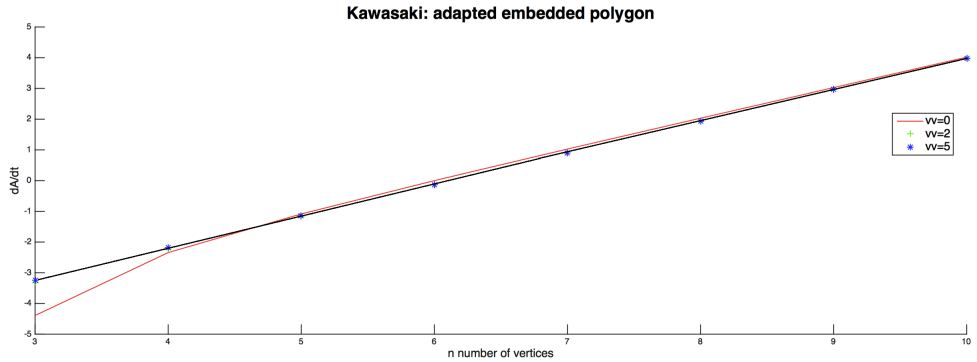


Figure 4.11: Change in area of an embedded polygon with virtual vertices ($vv = 0, vv = 2, vv = 5$) on a straight line connecting the triples. The black line is the Neumann-Mullins relation

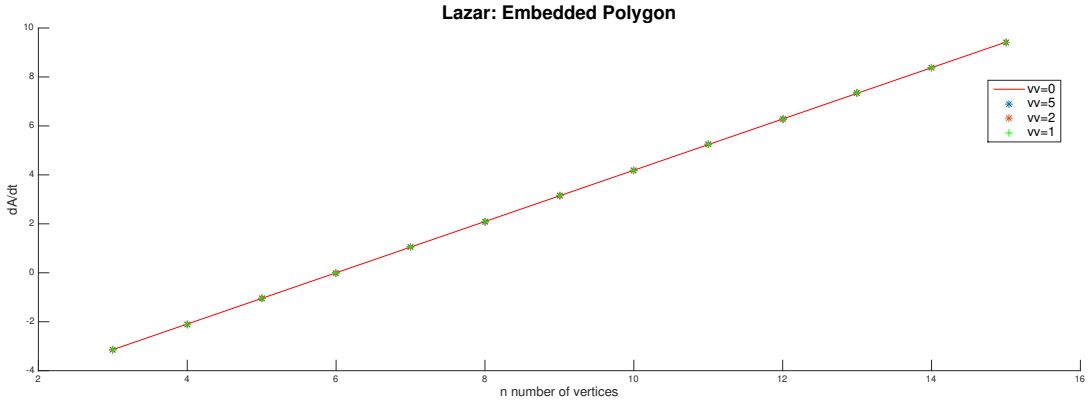


Figure 4.12: Change in area of an embedded polygon with virtual vertices ($vv = 0, vv = 1, vv = 2, vv = 5$) on a circle connecting the triples

To illustrate what happens two figures are made of a polygon with six triple points and four virtual vertices, figures 4.13 and 4.14. Notice that the virtual points are not anymore nicely distributed over the grain boundary edge. Concluded can be that this is due to the connection of a triple point and virtual point which gives an error in the solution. This occurs because the velocity of neighbouring cells is also taken in equation (4.4). In contrast to Kawasaki the method of Lazar makes sure that the virtual points stay equidistant and the triple points move enough outside to enlarge the area which is decreased by the virtual points. This effect is not anymore present when the virtual points are given on the straight line see figures 4.15, 4.16. An overview of the obtained results is given in table 4.1. Finally we can conclude that the method by Lazar is most suitable to describe grain growth by curvature.

Table 4.1: Overview of the result of the selected methods on the concerning test cases, where vv means number of virtual vertices

	Simple Polygon		Embedded Polygon Neumann-Mullins	
	Curvature	Neumann-Mullins	Arbitrary n, $vv = 0$	Arbitrary n, $vv > 0$
Nippon	+	-	-	+/-
Kawasaki	+	-	+	-
Lazar	+	+	+	+

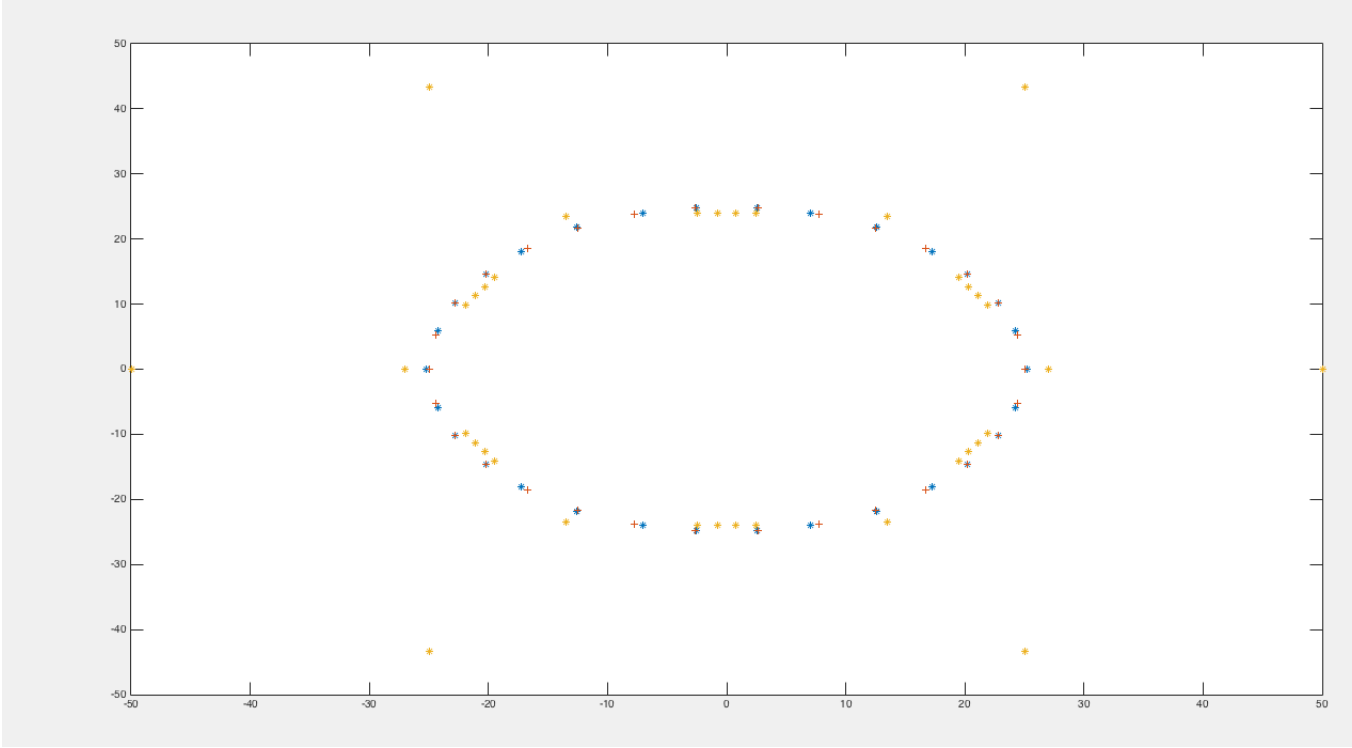


Figure 4.13: Kawasaki's method implemented on an embedded polygon with six triples and four virtual vertices, virtual vertices on the same circle. + at $t = 0$, blue * at $t = 1$, yellow * at $t = 10$.

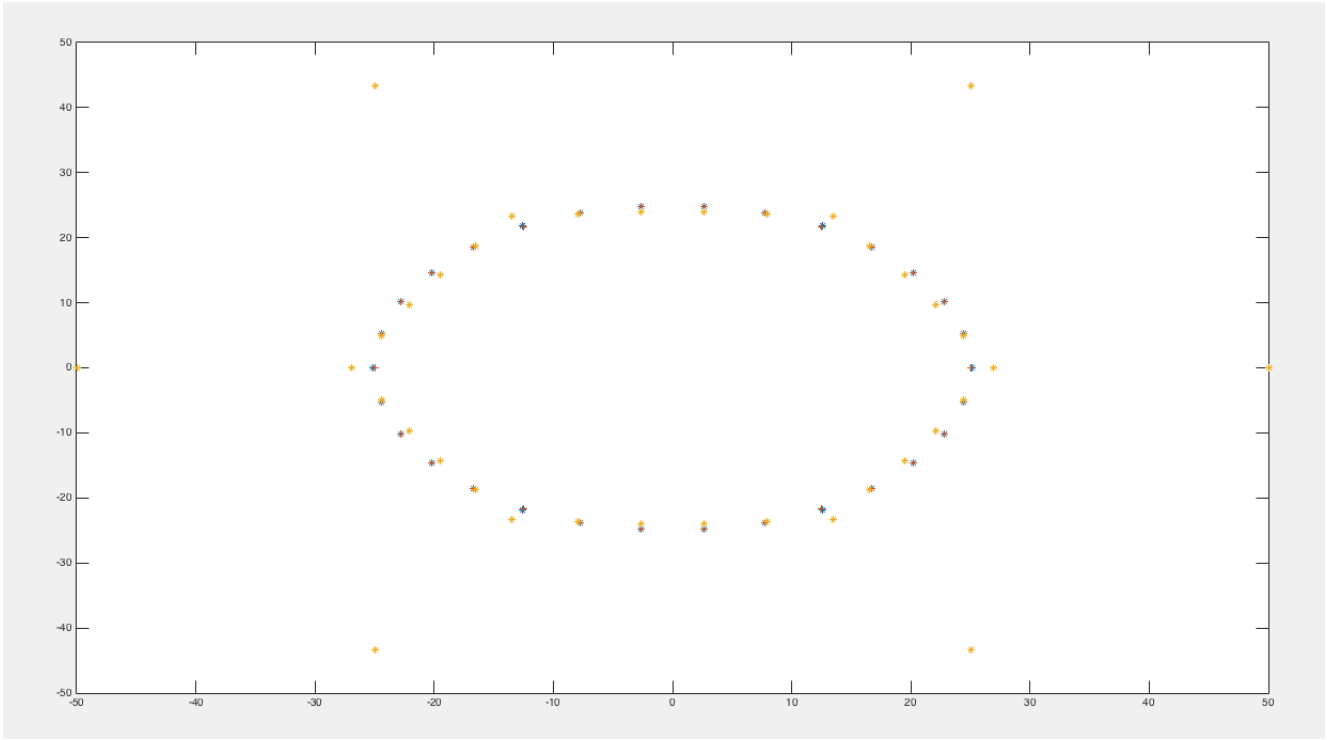


Figure 4.14: Lazar's method implemented on an embedded polygon with six triples and four virtual vertices, virtuals on the same circle. + at $t = 0$, blue * at $t = 1$, yellow * at $t = 10$.

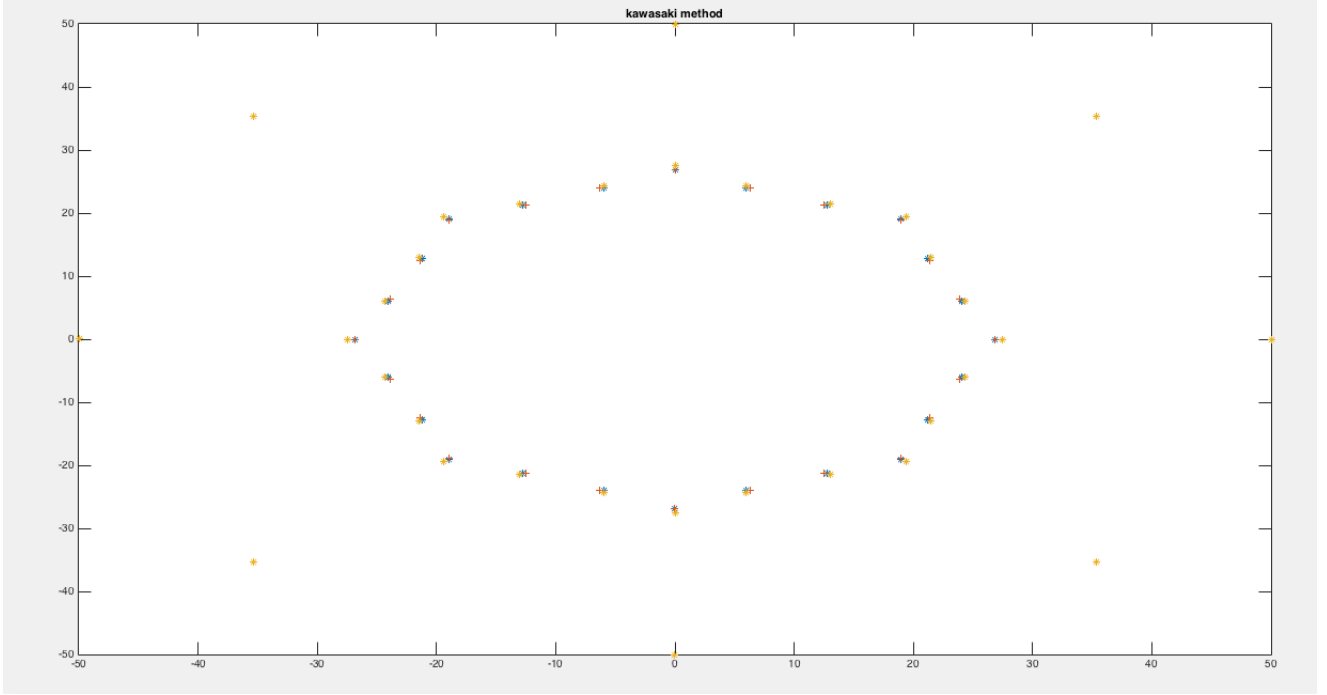


Figure 4.15: Kawasaki's method implemented on an embedded polygon with six triples and four virtual vertices, virtuals on the a straight line connecting the triples. + at $t = 0$, blue * at $t = 1$, yellow * at $t = 10$.

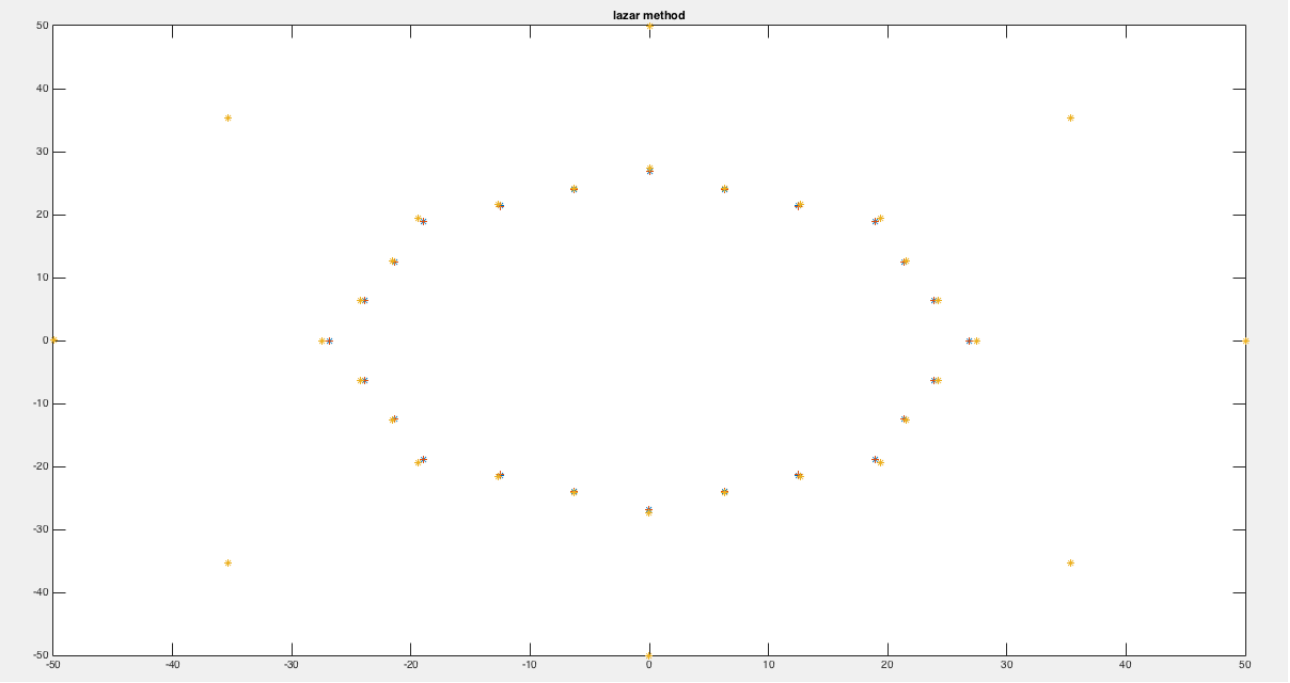


Figure 4.16: Lazar's method implemented on an embedded polygon with six triples and four virtual vertices, virtuals on the a straight line connecting the triples. + at $t = 0$, blue * at $t = 1$, yellow * at $t = 10$.

5

IMPLEMENTATION IN CELLULAR AUTOMATA

Finally the general model can be expanded to include grain growth by curvature. The procedures that are needed to be able to get a working hybrid Cellular Automata-Lazar method are presented. First the vertices (both virtual as triple) need to be extracted from the Cellular Automata model. Second the vertices will move according to the derived equations from Lazar (4.23). Finally all changes need to be updated back into the model.

5.1. FIRST PART – EXTRACTION

In this first part all the necessary information needed for the Lazar method is extracted. There are two different vertices that are needed for the description of the movement of a grain. So called triple points that lie between three grains and virtual vertices that lie on the edges of a grain; only having two mutual grains. First the triple points need to be extracted. Search through the interface cells and check whether more than three different grains lie in its Moore neighborhood. Interface cells are normal cells but with a tag such that they are recognized easily. The cell get this tag when in its Moore neighborhood two different states (grains) occur e.g. it lies on the interface. This limits the amount of cells needed to be evaluated enormously.

If a cell is found with three or four different grains in its neighborhood, it is again searched in a smaller neighborhood to find its exact location and can be added to the vertex list. All these vertices carry information about its exact and discrete location, connections and to which grains it belongs. Because our initial Voronoi is discrete it can rarely happen that four grains meet, in the analytic case the probability of this phenomena is zero. If such a vertex is found it will automatically be split into two vertices, dividing the four grains over the two.

The triple points give a good description of a grain. It not only gives the location of the end points of its edges, it also tells something about the average slope between the points, that is related to the curvature. When the interface between two vertices is a straight line the interface will then be accurately described. Unfortunately this is not the case, because the grain growth part will step in when other processes in the microstructure have stopped. It is not guaranteed to only have straight lines between the triples; that initially came from the starting Voronoi. To be able to follow the interface correctly additional vertices: virtual vertices are placed.

The algorithm works as follows:

1. Find the midpoint between two vertices
2. Depending on the slope of the edge look either horizontally or vertically, to find two different grains next to each other.
3. Use this location to calculate the Total Variation of second order [35]

$$TV(x) = \sum x_{i-1} + 2x_i - x_{i+1} \tag{5.1}$$

4. When this total variation is larger than a certain lower boundary an extra vertex is added
5. When new vertex is added two new edges appear and the same procedures needs to be repeated

When finished all grains will accurately be described by use of both triple points and virtual vertices.

Before the method by Lazar can be used some additional information is necessary. Finding the right connections between all triple points, sorting all vertices by grain and finally separate the interface cells by edge. The following information is now known:

- mvVertexList included both virtual and triples, position, grain-ids, connectionList
- mvGrainList, by grain sorted list with all vertices sorted by order. The list is needed to be able to track the size of the grains at all times, the vertices will be sorted by grain and in a way such that all vertices can be obtained in clockwise order.
- mvEdgeMap, by edge sorted interface cells. After moving the vertices limited to only one cell in either vertical or horizontal direction, also the cells at the interface need to be updated. Therefore the interface cells of the specific edges belonging to the triples are needed. A map container is used: every triple pair belonging to an edge is stored as a key and all interface cells belonging to this edge are stored as the mapped value.

5.2. SECOND PART – LAZAR METHOD

All the needed information from the Cellular Automata model is extracted, now grains are moved to minimize its grain boundary energy. In figure 5.1 an overview is found of the model. Assumed here is that initialization has finished, e.g. only the second and third part of the model is given. Some short explanations will be given on some of the functions used. Notice that the lower part of the figure belongs to part three of the model; placing back the moved vertices in the Cellular Automata model. In general be aware that every change or function needs to take in account the periodic boundaries.

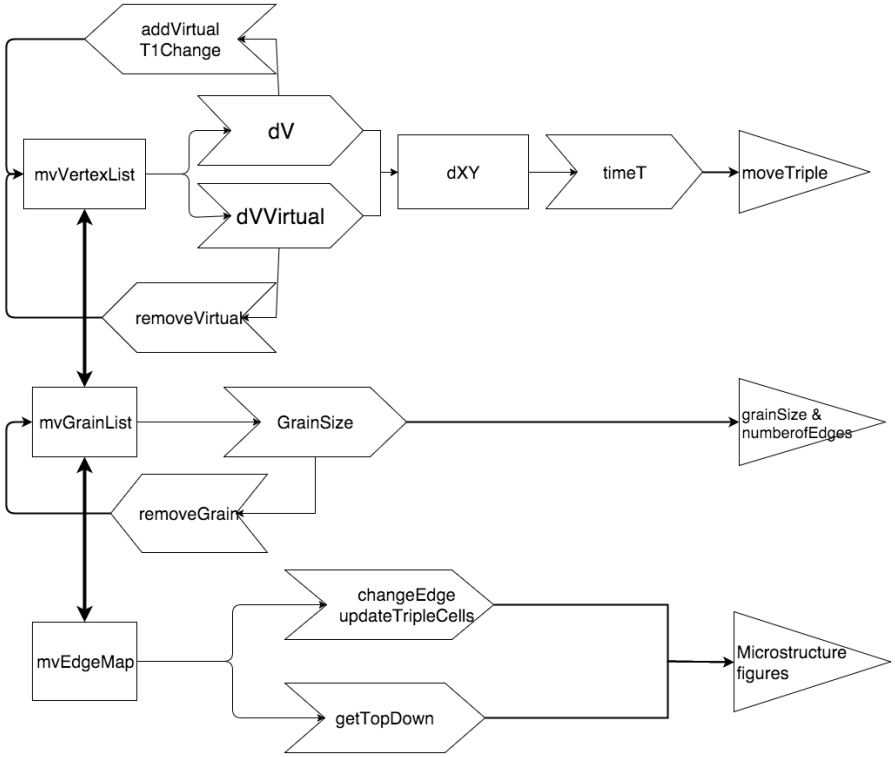


Figure 5.1: Overview of algorithm, rectangle is a data element, arrow shaped polygon is a function and the triangle is output

Start at the top of figure 5.1. From mvVertexList all essential information needed to calculate the velocities of the vertices can be obtained. This is calculated by using dV for the triple points and dVVirtual for the virtual vertices. Next is the variable vDx or vDy that keeps track of the movement in previous time steps. The reason that the coordinates of the triple points are not directly updated is that the displacement of the vertices needs to be restricted. Otherwise the procedure to place back the information into the CA model will be obstructed. Therefore the TimeStepFunction is used.

TIME STEP FUNCTION

The movement of a vertex is restricted to only one cell in horizontal direction and one cell in a vertical direction. Necessary as only the interface cells will be updated for every step. This restriction prevents a cell not being checked when the time step is too large and movement of a triple thus exceeds the interface.

Time t is obtained by calculating all velocities and making sure for every point that:

$$dX + t_{\max} v_x \leq \sqrt{2} \tag{5.2}$$

$$dY - t_{\max} v_y \leq \sqrt{2}, \tag{5.3}$$

where dX and dY is the total displacement of a triple of the previous time steps, but after its last use of moveTriple. The vertex for which dX or dY has reached $\sqrt{2}$ (the size of a cell is 1 by 1 such that the length scale is 1) will update the states of the cells surrounding the moved vertex and its edges, e.g. the third part of the model. The dX or dY of the considered vertex will be set to zero.

Within the process of the moving triples two procedures can occur. The removal or addition of vertices and a topological T1 change.

REMOVAL/ADDITION OF VIRTUAL VERTICES

In the original paper of Kawasaki [36] where the first two dimensional vertex model was proposed, only triple points were used. Further development in three dimensions revealed that more information of the edges is required. Therefore, a second kind of vertex is introduced: the virtual vertex [34]. These new introduced vertices are placed between a pair of triple points (called 'real vertices'), to allow a finer grid over the edges.

Define $n_{virtual}$ as the number of virtual vertices between two triple points, the precision of the discretization and therefore the precision of the approximation of the curvature depends on this number. A second parameter needs to be introduced, the minimal distance between two virtual vertices for which otherwise one vertex needs to be removed. Due to grain growth the structure will change, hence the number of virtual vertices need to depend on the mean grain size $< r >$. Therefore, the minimum distance, Δ between virtual vertices is given [36]:

$$\Delta = \frac{\alpha}{n_{virtual} + 1} < r >, \tag{5.4}$$

where $< r >= [(2A_{total})/(\pi n_{real})]^{\frac{1}{2}}$ using Eulers relation in two dimensions, A_{total} the size of the system, n_{real} the number of triple points in the system. The pre-factor α is chosen small enough (0.025 in Weygand et al. [34]) to have negligible influence on the size distribution of the grains. When the distance between two real vertices is larger than $2.5 n_{virtual}\Delta$, $n_{virtual}$ virtual vertices are introduced.

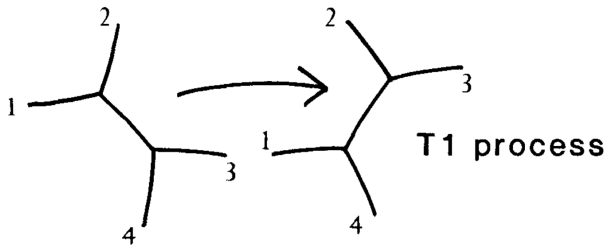
TOPOLOGICAL CHANGES

Next to the movement of vertices according to the equations of motions defined in 4.23 , some topological changes need to be enforced. In the two-dimensional case the following needs to be considered:

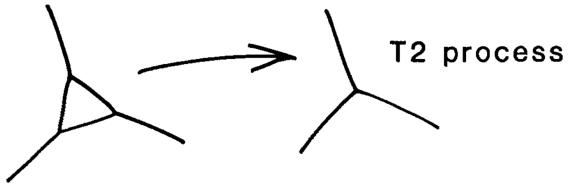
1. removal/addition of virtual vertices
2. change in connection
3. removal of a grain

CONNECTION CHANGE & REMOVAL OF A GRAIN

The first topological transformation (seen in figure 5.2a) occurs when two triple points are in close distance from each and get smaller. Then the vertex connections will change as illustrated. A consequence of this transformation is that two grains will lose an edge and two grains will gain an edge. Changes will have to be



(a) Transformation 1



(b) Transformation 2

Figure 5.2: Topological transformations, source: [34]

made in mvVertexList, mvGrainList and mvEdgeMap.

The second topological transformation (seen in figure 5.2b) removes two triple point when a three sided grain which has one side smaller than Δ and the other sides smaller than 2Δ. In this case the size of the grain is smaller than Δ^2. Equivalent, remove a grain when its size is smaller than a certain parameter. To be able to calculate the grain size use Green's Theorem, to give an expression for the total area of a closed curve:

$$A = \oint_c x dy \tag{5.5}$$

Parametrize each segment of the polygon from (x_k, y_k) to (x_{k+1}, y_{k+1}):

$$C_k : \mathbf{r} = ((x_{k+1} - x_k)t + x_k, (y_{k+1} - y_k)t + y_k) \text{ with } 0 \leq t \leq 1 \tag{5.6}$$

Plugging in gives the equation of the area:

$$A = \sum_{k=0}^{n} \frac{(x_{k+1} + x_k)(y_{k+1} - y_k)}{2} \tag{5.7}$$

5.3. THIRD PART – UPDATING THE CELLULAR AUTOMATA MODEL

To really obtain a hybrid CA-Lazar model, the information found using Lazar's model needs to be placed back into the CA model. This is done by making sure that the vertices enclosing a grain corresponds to the grain found when looking at the states of the cells in the CA model. When a vertex moves, of course its edges implying the borders of a grain will move and therefore the states telling whether the cell belongs to grain A or grain B at the interface.

In this third part there are two important functions making sure that the right cells will be updated when a vertex has been moved. One function only updates the interface cells at the edges where the number of different states (grains) is limited to two. The other focuses on the cells near a triple point where the number of neighbors is equal to three.

Looking specifically at the interface cells of the edge, the decision whether a cell lies within a grain is made by checking whether it lies above or under the straight line connecting the two vertices of this edge. Needed now is the information which grain lies above or under this line. The function getTopDown will give this information. It uses the two vectors belonging to a triple, where one belongs to the changed edge, the other is one of the remaining two where the angle between the two vectors is not larger than 180 degrees. It next calculates the dot product and projects this on the vector belonging to the edge of the considered triples. The orthogonal component will point to which side of the edge the considered grain lays, who have both edges in common.

When all three edges belonging to the moved triple are evaluated, only the interface cells with more than two grains in their neighborhood need to be checked. This is done by updateTripleCells, that looks in the neighborhood of the triple for these kind of cells. Next is to evaluate the angle of the vector that starts at the vertex and ends in the middle of the considered cell. Comparing this angle with the angles of the edges the corresponding grain can be found.

6

RESULTS FROM HYBRID CA-LAZAR MODEL

Before a general example will be given, it will be shown that some essential parts of the model work correctly. The first part is extraction, it can be concluded that all vertices are successfully extracted as no problems arise when the triples get in motion. Considering the second part, some special cases will be shown to validate of the model. The third part has also been performed and will be discussed.

6.1. SPECIAL CASES CONSIDERED

Before the result of the movement of all vertices in the 2D case is given, two special cases will be examined: the removal of a grain and a T1 change.

In Chapter 4 it can be observed that grains with less than six edges reduce in size and the ones with more edges grow. In the process of getting smaller the grain loses triple points by a T1 change. This process can be followed in figure 6.1 and 6.2. A T1 change occurs when two triple points move to each other. If a grain has come to the point that it only contains three triple points and is small enough the grain will be removed. This process can be followed in figure 6.3 and 6.4.

To follow more closely the effects in these special cases, a function has been written to stop all grains containing more than three or four edges from moving. The grains are shown at specific interesting time steps, note that this does not tell anything about real time as this differs due to the function timestep.

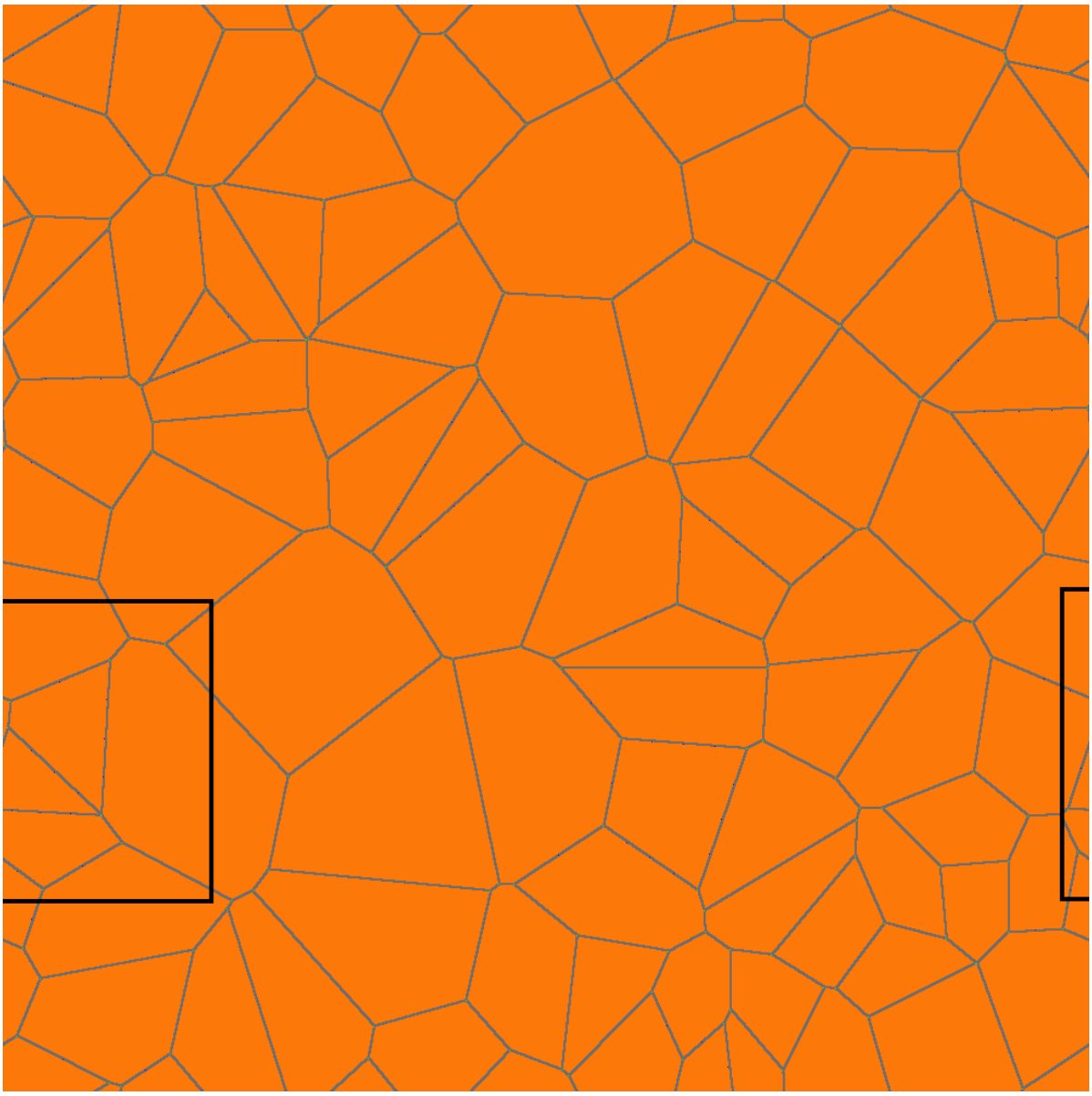
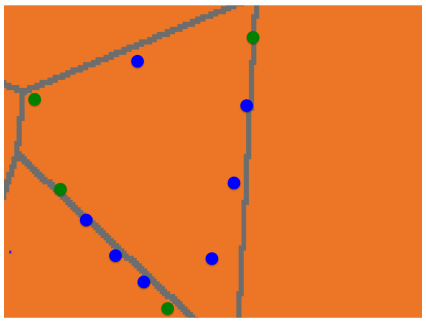
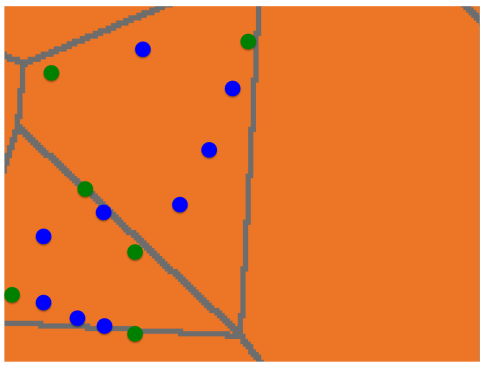


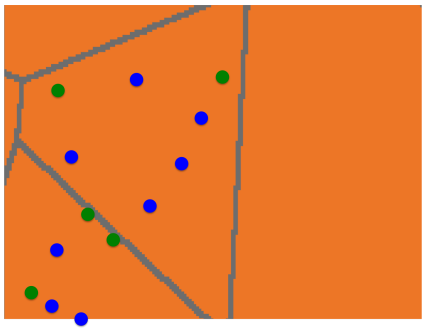
Figure 6.1: A voronoi with a polygon having four edges. The green dot is a triple point, the blue dots are virtual vertices



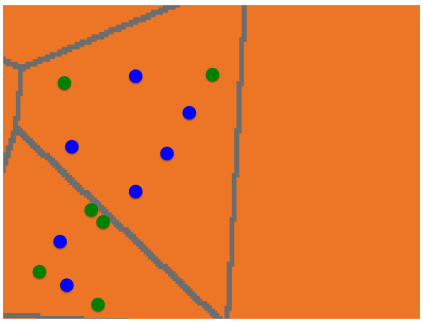
(a) Two polygon with each four edges are moving. Especially notice two triples moving towards each other. T=141



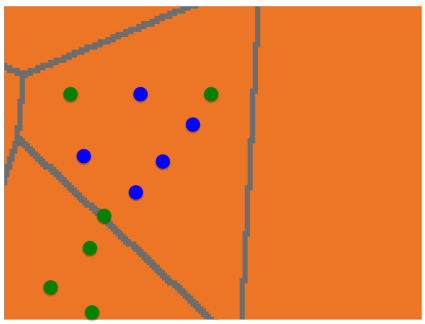
(b) Two polygon with each four edges are moving. Especially notice two triples moving towards each other, where two virtual vertices have been removed. T=243



(c) Two polygon with each four edges are moving. Especially notice two triples moving towards each other, last virtual is removed. T=301



(d) Two polygon with each four edges are moving. T1-change has been executed. T=332



(e) Two polygon with each four edges are moving. Now the two grains have both only three edges. T=367

Figure 6.2: T1 change of a polygon with four edges. The green dot is a triple point, the blue dots are virtual vertices

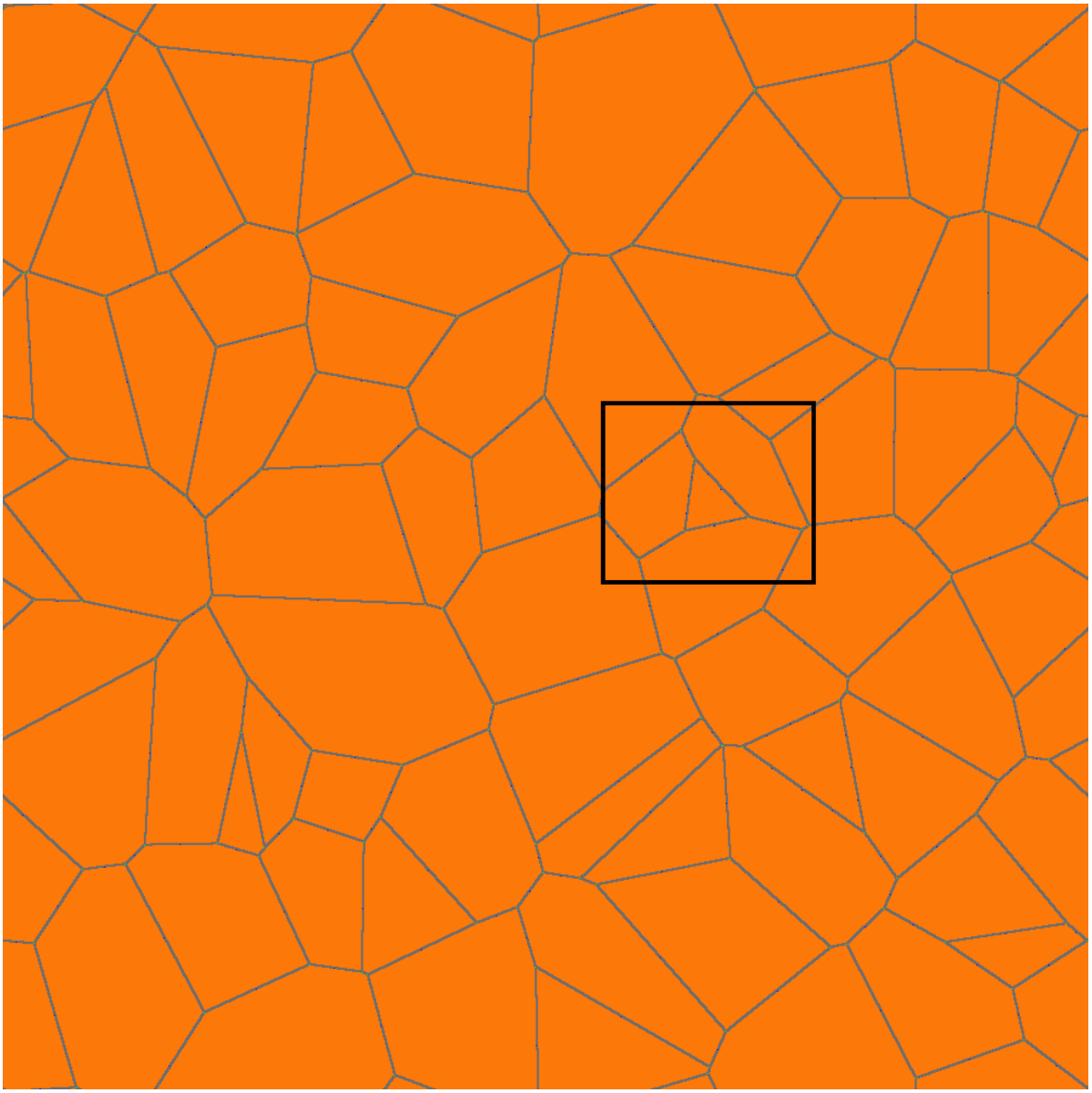


Figure 6.3: A voronoi with a polygon having three edges. The green dot is a triple point the blue dots are virtual vertices

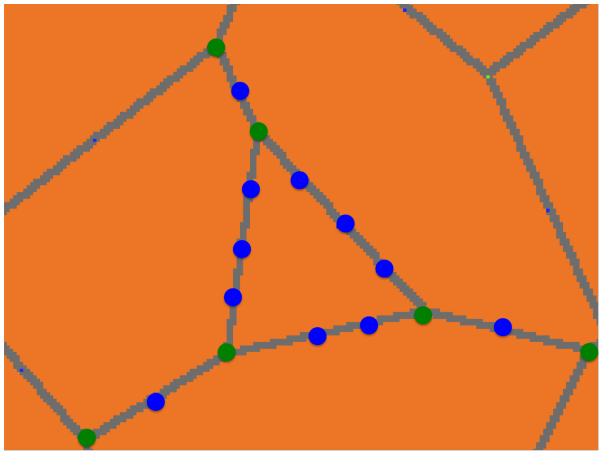
6.2. PLACEMENT OF VIRTUAL VERTICES

According to [36] no additional virtual vertices are needed in the 2D case. This is in contradiction with the results from this model and the observation of Lazar [4]. First of all there is the need to follow the grain boundary more accurately. An algorithm has been developed to place additional vertices. It places virtual vertices correctly on the interface, but has not been tested for all cases. Next it can be observed that a vertex is not moving correctly if not enough virtual vertices are placed. Consider the following case:

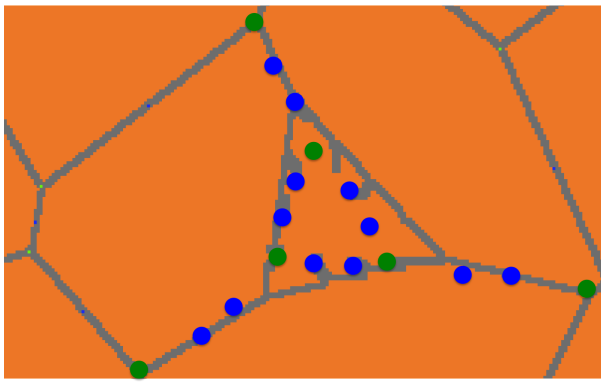
A triple connected to three other vertices by the vectors:

$$\begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 10 \\ 10 \end{pmatrix}. \tag{6.1}$$

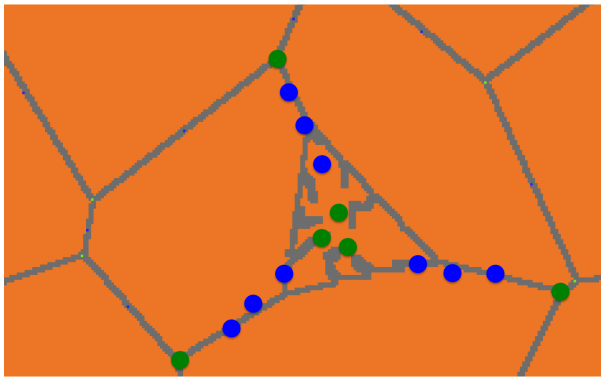
Calculating the velocity vector for this triple point gives:



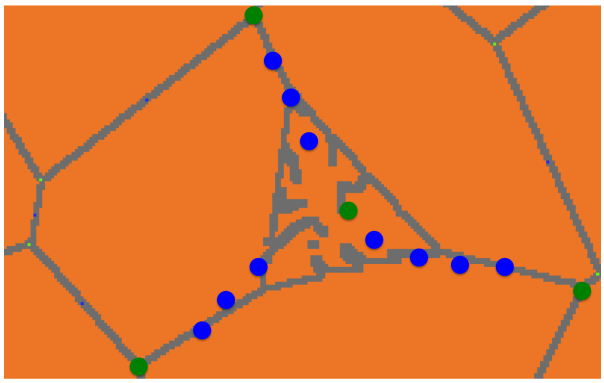
(a) Zoomed in at the specific polygon at T=0



(b) Triples move to center at timestep T = 99



(c) Grain is getting smaller almost small enough to perform change, T = 177



(d) Removal of grain is performed T=188

Figure 6.4: Removal of a grain with three edges. The green dot is a triple point, the blue dots are virtual vertices

$$d\mathbf{v} = \begin{pmatrix} -52.36 \\ -42.34 \end{pmatrix}. \tag{6.2}$$

If a virtual was added to the largest vector (edge), the length of the vector belonging to triple point is reduced. Therefore replace $\begin{pmatrix} 10 \\ 10 \end{pmatrix}$ with a smaller vector:

$$\begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{6.3}$$

Calculating the velocity vector for this triple point again gives:

$$d\mathbf{v} = \begin{pmatrix} -52.36 \\ 104.72 \end{pmatrix}. \tag{6.4}$$

This illustrates the important effect the number of virtual vertices have on the velocity vector of a triple point. To prevent this from happening more virtual vertices need to be introduced. Now only a certain lower bound is used, but a more dynamic description of the necessary amount of virtual vertices is preferred.

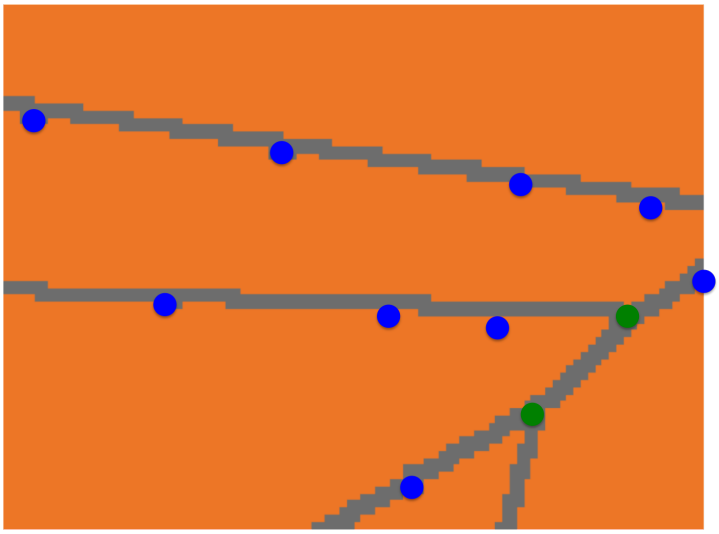
6.3. UPDATING INFORMATION BACK INTO CA

For long the method to update the information back into the Cellular Automata did not function. Now it works but there are still improvements necessary. These improvements are needed in the case virtual vertices are used and or topological changes are enforced.

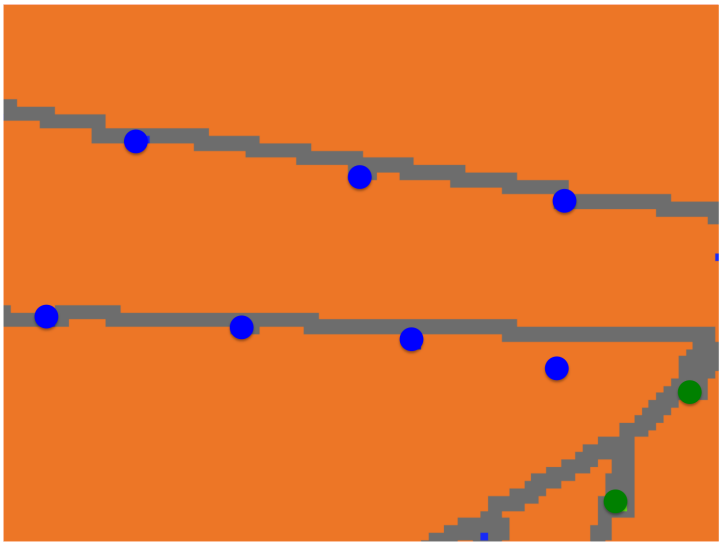
6.4. BEHAVIOUR OF ALL GRAINS: EXAMPLE

After considering special cases, now a general example of the movement of all triples under the influence of minimization of grain boundary energy. A Voronoi is made for a grid size of 1000x1000 containing 70 different grains is given. All triple points are found and where needed virtual vertices placed.

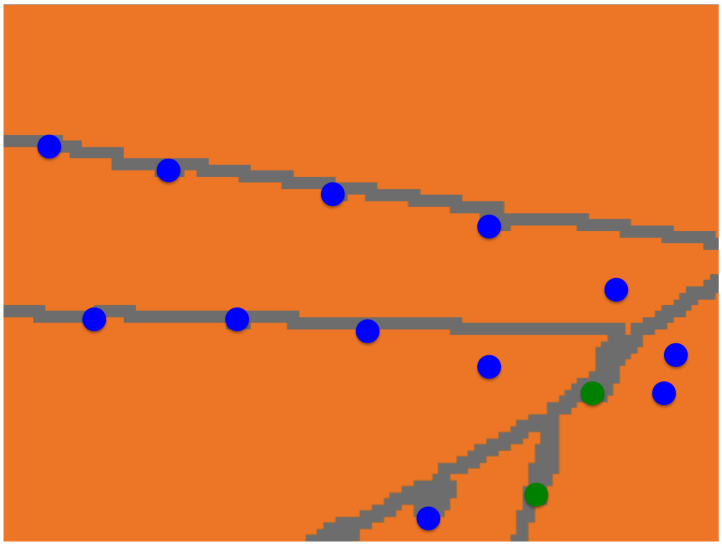
The model behaves as expected. In figure 6.6a it can be observed that grains with less than six triples reduce in size, with six are stable in size and more than six grow in size. Next it can be seen that the rate of growth is also proportional to the amount of triples a grain has. Around time $t = 18$ the straight line is irregular, this is due to topological changes. A grain with for example 7 triples reduces to 6 and is in general of a higher average size, therefore increasing the average grain size of grains with 6 triples. The next two figures (6.6b, 6.6c) show how the distribution of size develops over time. In the starting situation, it can be seen that the bulk of grains are close to the average grain size. As the process of minimization of grain boundary energy develops, small grains get reduced in size even more and later on disappear, whereas large grains will tend to get even bigger. This is the case for as well the size of grains, as the amount of triples.



(a) Zoomed in at the specific polygon at T=2

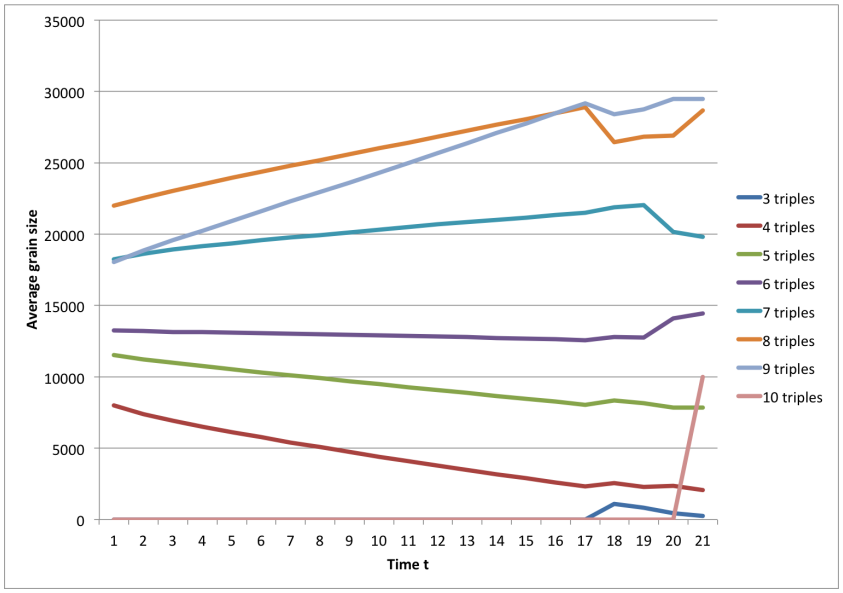


(b) Unstable vertex moves in such a way that the grain becomes larger at timestep T = 64

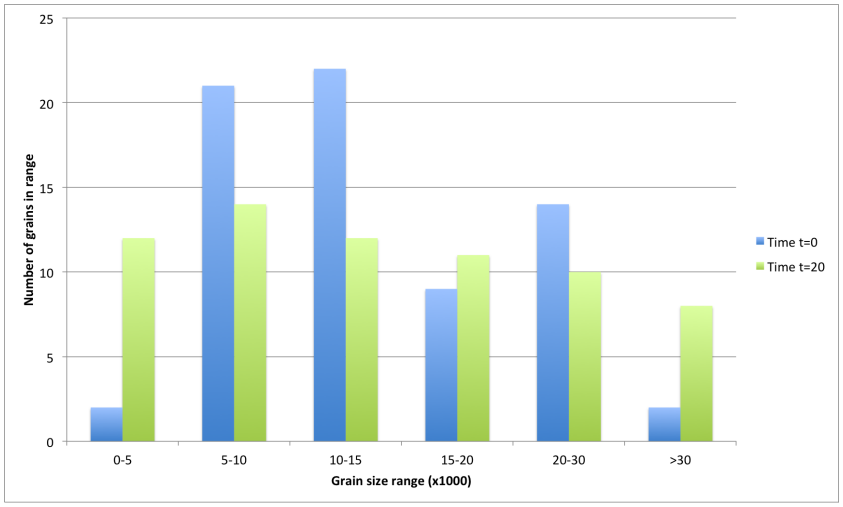


(c) Unstable vertex moves in such a way that the grain becomes larger, T = 100

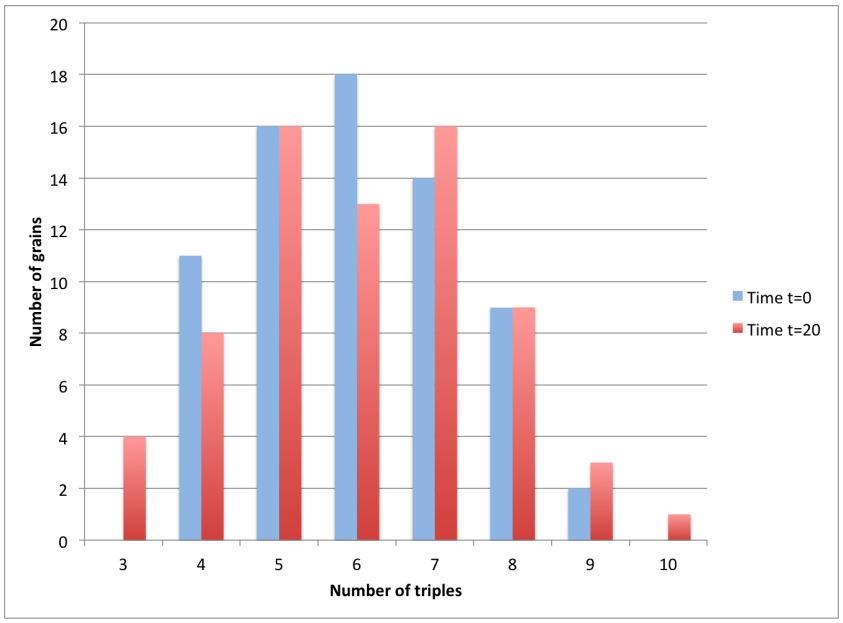
Figure 6.5: A polygon with an unstable vertex.The green dot is a triple point, the blue dots are virtual vertices



(a) Here we see the average grainsize develop over time grains with different number of triples



(b) A grainsize distribution, the number of grains that are in a certain grain size range



(c) Distribution of grains with a certain number of triples (edges)

Figure 6.6: An example of the development of grains due to minimization of grain boundary energy

7

FIRST CONCLUSIONS & FURTHER RESEARCH

Vertex based methods have been selected to model grain growth. These methods are preferred over counting cell methods and other derived counting cell methods for the following two reasons. First, an important factor is the computation time for a counting cell method. Instead of using only a few points at the intersection of grains (vertices), for all interface cells the curvature needs to be calculated. To be able to calculate the curvature the number of cells in a neighborhood belonging to grain A or grain B are counted. When the curvature is very small the size of the neighborhood needs to be enlarged to get the necessary accuracy. This variation in size of the neighborhood will add even more complexity to the model.

Three vertex methods have been tested: Tamaki [31], Weygand et al. [34] and Lazar [4]. Concluding from the results of the polygon test cases, Lazars method has been chosen to further implement in a 2D structure.

A hybrid cellular automata-Lazar model has been constructed. Summarizing:

1. Extract the necessary data (vertices) from the CA model.
2. The vertices will move according to the derived equations from Lazar (4.23).
3. Update all changes back into the CA model.

It can be concluded that virtual vertices are crucial in the extraction of information from the CA model. To describe the interface, as well as to prevent the effect of large differences in the length of vectors needed for the calculation of the velocity vector of a triple point.

From the results it can be concluded that the model works correctly. Considering special cases, it can be observed that both topological changes work, since:

1. Small grains with only three edges have been removed.
2. A correct change of connections when two vertices are close to each other.

In the general example we have seen that larger grains grow in favor of the smaller grains, similar to the phenomenon of Oswald ripening.

Although updating the changes back into the CA model works in the most basic case, still some improvements need to be made here. The model needs to be adjusted such that it also can handle topological changes and the addition and/or removal of virtual vertices.

8

APPENDIX ALL ADDED FUNCTIONS WITH SHORT DESCRIPTION

```
1
2  class Global: public TThreadingBasic
3  {
4  private:
5    TvertexList mvVertexVector; //!<List to store vertices for grain growth
6    TvertexList mvGrainList; //!<List to store the grains
7    TvertexList mvVertexVector_quad;//!<List to store of 4 neighbour_id vertices for grain growth
8    std::vector<vertexPair> vertexPairVector;//??????
9    TedgeMap mvEdgeMap;
10   void grow_gg_bastiaan(TICellsContainer &icells,TSubSystem &SubSystem, TvertexList &mvVertexVector,
       TedgeMap &mvEdgeMap); //!<new grow function by bastiaan..
11     void ProcessNewChanged(TSubSystem &sub); //!< deal with newly transformed cells
12   void findVertices();
13   void findVirtualVertices();
14   void findGrain(int Nx);
15   double solveEquationt(double d_xy, double dv_xy);
16 //     void findEdges(TICellsContainer &icells);
17 };
18 class Cells: public TThreadingBasic
19 {
20 private:
21   int mvNNearest; //!< number of nearest neighbours, 2D: 4, 3D: 6
22   int mvNNext;  //!< number of next nearest neighbours, 2D: 4, 3D: 12
23   int mvNNextNext; //!< number of next-next nearest neighbours, 2D: 0, 3D: 8
24   int mvNCube;//!< number of neighbours in 2D 3
25   int mvNCubeL;//!<number of neighbours in 2D 3
26   void updateTripleCells(int direction, int tripleList_index, int index, TvertexList &mvVertexVector,
       TSubSystem &SubSystem, TedgeMap &mvEdgeMap);//!<function that
27   void AddNewChangedCell(TSubSystem &sub,TCellChangeNew &ccn)const; //!< adds a to be changed cell to the
       list
28   std::pair<double,double> findVirtualVertex(std::pair<double,double> pair1, std::pair<double,double>
       pair2, double s); //!< finds virtual vertices to follow the interface of the grain more accurately
29   void changeEdge(int tripleList_index, int index, TvertexList &mvVertexVector, TSubSystem &SubSystem,
       TedgeMap &mvEdgeMap); //!<Updates the the edges belonging to the moved triple
30   int nbrSizef(int index); //!< Returns the nbrsize of a vertex
31 };
```

```
32
33 class Tvertex{
34
35 public:
36   double vXVertex; //!< Exact x coordinate of a vertex
37   double vYVertex; //!< Exact y coordinate of a vertex
38   double vDx; //!< Total x displacement before updating a triple, will be again zero when total
       displacement> sqrt(2)
39   double vDy; //!< Total y displacement before updating a triple, will be again zero when total
       displacement> sqrt(2)
40   int T1index; //!< Keeps track of the usage of a t1change
41   int vi;  //!< i coordinate of cell belonging to vertex/triple point
42   int vj; //!< j coordinate of cell belonging to vertex/triple point
43   std::set<int> vNeighbours;//<! set of all grain_id's of the cells in a Neumann neigbhourhood
44   std::vector<int> mvVertexConnect; //<! vector with the triples connected to the considered triple
45   std::vector<int> mvQuadList; //<! List of quadrupple points who need to be removed
46   Tvertex(); //!< default constructor
47   void removeConnect(); //!<
48   void addNeighbours(const std::set<int> &neighbours); //!< add a grain_id to the set of neighbours
49   void addConnect(int i); //!< add a triple_id to the list of connections
50   void clearConnect(); //!< removes all connections stored in mvvertexconnect
51   std::vector<int> mvVertexDIFFC;
52   int viDiag; //!< i coordinate of diagonal neighbour cell of cell i,j,k (also belonging to vertex/triple
       point)
53   int vjDiag;
54   int vertexIndex;
55   double timeT;
56
57 };
58
59 class TedgeMap{
60 private:
61   std::map< std::pair<int,int>, std::set<int> > mvEdgeMap; //!< buiding a map container where all
       connected triple pairs are the keys mapped to all interface cell belonging to the specific edge
62
63 public:
64
65   void addEdgeMap(std::pair<int,int> edgePair, std::set<int> edgePairVector); //!< add the key with mapped
       indices to mvEdgeMap
66   std::set<int> getEdgeVector(std::pair<int,int> triplePair); //!< get the set of indices belonging to the
       specific key
67   void removeKey(std::pair<int,int> triplePair); //!< remove the key from the edgemap
68   std::vector<std::pair<int,int>> getPairVector();
69 };
70
71
72 class TvertexList{
73
74 private:
75   std::vector<Tvertex> mvVertexVector; //!< a vector of Tvertex
76   std::vector<Tvertex> mvGrainList; //!< per Tvertex their is one grain with all vertices belonging to the
       grain listed clockwise
77
78
79   void removeTriple(int i); //!< removes a vertex in mvVertexVector
80   void removeQuad(int i); //!< removes a quad in mvVertexVector
81
82 public:
83   void changeNeighbours(int i, std::set<int> newNeighbours); //!< changes the neighbours grain-ids with a
```

```
    new set of neighbours
84  void addVertex(const Tvertex &v); //!< add a vertex to vertexvector
85  int getConnect(int i, int j); //!< gets the connected triple of triple i on location j in
      mvVertexConnect
86  void addGrain(const Tvertex &v); //!< adds a grain to mvGrainList
87  bool inGrainsize3(int a); //!< checks whether triple a belongs to a grain with only three triple points
88
89  void findConnection(); //!< Searches through mvVertexVector and checks whether a triple has a neighbour
      combination in common, when so this is added to the connectionlist of both triples
90  //  void cleanForQuads(); //!<
91  int countQuads(int i); //!< counts the number of quadruples
92  bool addVirtual(std::vector<double> e1, int i, int a, TvertexList &mvGrainList, int Nx); //!< When a
      vector from triple becomes to long an extra virtual vertex needs to be added to both mvVertexList and
       mvGrainList
93  bool removeVirtual(int a, TvertexList &mvGrainList); //!<When two vertices (where one is a virtual
      vertex) come to near to each other the virtual vertex needs to be removed
94  bool VertexInList(const Tvertex &v); // Check whether the vertex is already in the mvVertexList
95  void updateTriple(); //not used anymore
96  void updateTriple1(); //not used anymore
97  int size(); // returns size of vector
98  int listSize();  // returns size of grainlist
99  int numberOfGrain();   // returns size of grainlist
100 int grainConnectSize(int i);  // returns size of grain
101 Tvertex getVertex(int i); //returns vertex
102 void getIJForTriple(int i, int &vi,int &vj); //returns coordinates of vertex
103 void printVertex(int i); //print vertex
104 std::vector<double> getTripleVector(int i, int j, int Nx);//!< returns a vector which originates in
      triple i and goes to its connectionList[0,1,2]
105 std::pair<double,double> getTripleVector1(int i, int j, int Nx);//!< returns a pair which corresponds to
       a vector originating in triple i to triple j
106 //  std::vector<int> getTripleOrder(int i, int Nx, int Ny); //!<
107 //  std::vector<int> getGrainidOrder(int i,std::vector<int> vOrder); //!< uitleg
108 double getDistance(std::vector<double> tripleVector); //!<returns the length of a vector
109 std::vector<int> getBeginEnd(int i, int j); //!< returns the coordinates of the begin and endpoint of
      the edge
110 void moveTriple_x(int i, double dx, double tmin, int Nx); //!< moves a triple in x direction taking in
      account the periodic boundaries (exact coordinates)
111 void moveTriple_y(int i, double dy, double tmin, int Nx); //!< moves a triple in y direction taking in
      account the periodic boundaries (exact coordinates)
112 void moveTriple(int tripleList_index, int dx, int dy, int Nx); //!< moves a triple in both direction
      changes its integer coordinates
113 std::vector<double> dv(int i, int Nx, TvertexList &mvGrainList); //!< calculating the velocity of a
      triple using Lazar
114 std::vector<double> dvVirtual(int i, int Nx, TvertexList &mvGrainList); //!< calculating the velocity of
       a virtual point
115 void findGrain(); //!< searches through mvVertexList and sort the triples by grain using its
      neighbourlist
116 double getGrainSize(std::vector<int> GrainListLocal, int Nx); //!<Calculates the grain size using the
      derived formula
117 //  int indexConnectedTriple(int tripleList_index, int grain_id1, int grain_id2, int Nx);
118 bool T1change(int a, int Nx, double dx, double dy,TvertexList &mvGrainList); //!< function enforces a
      topologica t1change
119 void resetPoint(int a, int b, int a1, int a2); //!< When a t1 change is excecuted the two changed
      triples need to change their coordinates
120 void updateConnectionList(int oldIndex, int newIndex, int a1, int a2); //!<when a vertex is added or
      removed the connectionlist needs to be updated
121 void replaceGrain(int i, std::vector<int> grainVector); //!< When a vertex is added or removed the
      connectionlist of a mvGrainList needs to be updated
122 std::vector<int> grainOrder(std::vector<int> collectVertex); //< After findGrain all vertices in the
```

```
    grainlist need to be sorted such that it is possible to go through the grain clockwise
123 bool isPeriodic(std::pair<int,int> vertexPair,int Nx); //!< Checks whether a vector uses the periodic
    boundary
124 bool isPeriodicX(std::pair<int,int> vertexPair, double x,int Nx); //!< Checks whether a vector uses the
    periodic boundary
125 bool isPeriodicY(std::pair<int,int> vertexPair, double y ,int Nx); //!< Checks whether a vector uses the
     periodic boundary
126 void moveDx(int i, double dv_x, double tmin); //!< changes its vDx by dv_x*tmin
127 void moveDy(int i, double dv_y, double tmin);//!< changes its vDy by dv_y*tmin
128 void resetDx(int i); //!< when a triple has the larges dv_x it needs to reset vDX
129 void resetDy(int i);//!< when a triple has the largest dv_y it needs to reset vDY
130 double getDx(int i); //!<returns vDx
131 double getDy(int i);//!< returns vDy
132 double getvXVertex(int i); //!< returns vXVertex
133 double getvYVertex(int i);//!< returns vYVertex
134 int getConnectSize(int i); //!< returns whether triple or virtual vertex
135 std::vector<int> getGrainListLocal(int i); //!< returns the list of vertices belonging to grain i
136 std::vector<int> getTripleGrainListLocal(std::vector<int> GrainListLocal); //!< returns a list of
    triples belonging to grain i
137 void clearGrainList(); //!< clears grainlist
138 std::pair<int,int> findTriplePairofEdge(int a, int b); //!<Input two grain-ids, returns triplepair
139 std::vector<std::pair<int,int>> ConstructCoord(int i, int j, int n, int Nx); //!<projects vector of
    triplepair such that the periodic boundary will not interfere
140 std::pair<int,int> neighboursOfPair(int i, int j); //!< returns the grain-ids of a vertexpair
141 std::pair<int,int> getConnectPair(int grain, int triple); //!< returns the two vertices connected to the
     triple and belonging to the same grain
142 double getS(int i, int j, int Nx); //!< Returns the slope of a vector belonging to an edge
143 std::vector<int> getCollectVertex(int k); //!< collects all vertices belonging to grain k
144 std::pair<int,int> getTopDown(double s, int connectTriple, int triple_index, int Nx); //!< Function
    which defines which grain-id is above/under or left/right of the vectorline connecting to vertices
145 void virtualConnect(int i, int j, int index); //!<connects the newly added virtual vertex to its
    neighbouring vertices
146 double angle(std::vector<double> e1, std::vector<double> e2); //!< returns the angle between the two
    vectors
147 int findVirtualofTriple(std::pair<int,int> edgeTriplePair); //!< returns the vertex which has a
    connection to both vertices of edgeTriplePair
148 void removeGrain(int a, int b, int c, TvertexList &mvGrainList, int grain);//!<removes the grain from
    mvGrainlist, mvVertexList
149 bool inGrainWithSize(int a, int sizeGrain, TvertexList &mvVertexVector); //!<returns true if the number
    of edges is equal to sizeGrain
150 int numberOfEdges(TvertexList &mvVertexVector, int a); //!< returns the number of edges of the grain
151 int pointInGrain(int tripleList_index, int indexPoint, int Nx); //!< using the angles of the vectors
    originating from the triple the function gives the grain-id the evaluated point belongs to
152 //  bool signOfVector(std::vector<double> vector1, std::vector<double> vector2, std::vector<double>
    pointVector); //!<
153 std::vector<double> getIndex2TripleVector(int i, int x, int y, int Nx); //!< Returns the vector between
    a triple andthe middle of a cell
154 int getGrainID(int i, int a, int b); //!<returns the mutual grain-id of three vertices
155 double angleCounterClockwise(std::vector<double> v1); //!<gives the angle between positive x-axis and a
    vector
156 double angleBetweenVectorsCC(std::vector<double> v1, std::vector<double> v2); //!<gives the angle
    between two vectors using atan2
157 double relativeAngle(double pointAngle, double vectorAngle);//!< returns the angle which adds 2pi when
    negative
158 bool pointInTriangle(std::vector<double> pointVector, std::vector<double> vector1, std::vector<double>
    vector2); //!<whether a point is inside a triangle
159 int testVectorTriangle(std::vector<double> v1, std::vector<double> v2, std::vector<double> v3,std::
    vector<double> p);//!<tests a point between which vectors it lies
160 void updateT(double deltaT ); //!< updates time T
```

```
161   double getTimeT(); ///< returns timeT
162 };
```

BIBLIOGRAPHY

[1] C.Bos, M. Mecozzi, and J. Sietsma, *A microstructure model for recrystallisation and phase transformation during the dual-phase steel annealing cycle,* Computational Materials Science , 692 (2010).

[2] B. F. van der Boor, *Grain growth by curvature, Literature study*, Ph.D. thesis, TU Delft (2015).

[3] E. a. Lazar, J. K. Mason, R. D. MacPherson, and D. J. Srolovitz, *A more accurate three-dimensional grain growth algorithm,* Acta Materialia , 6837 (2011).

[4] E. a. Lazar, R. D. MacPherson, and D. J. Srolovitz, *A more accurate two-dimensional grain growth algorithm,* Acta Materialia , 364 (2010).

[5] R. D. MacPherson and D. J. Srolovitz, *The von Neumann relation generalized to coarsening of three-dimensional microstructures.* Nature , 1053 (2007).

[6] D. a. Porter and K. E. Eastering, *Phase Transformations in Metals and Alloys,* , 540 (1992).

[7] *Calphad phase diagram,* http://www.calphad.com/iron-molybdenum.html, 29-07-2015.

[8] *Austenite and ferrite crystal structure,* https://en.wikipedia.org/wiki/Austenite, 29-07-2015.

[9] *Edge dislocation,* http://www2.warwick.ac.uk/fac/sci/physics/current/postgraduate/regs/mpags/ex5/strainedlayer/disloc1/, 29-07-2015.

[10] *Screw dislocation,* https://www.nde-ed.org/EducationResources/CommunityCollege/Materials/Structure/linear_defects.html, 29-07-2015.

[11] *Dislocation annihilated,* http://users.encs.concordia.ca/~mcqueen/Dislocations_lecture8.html, 29-07-2015.

[12] J. Czochralski, Int. Z. Metallogr. , 289 (1914).

[13] J. Ewing and W. Rosenhain, Phil. Trans. roy. Soc. A , 353 (1900).

[14] C. Benedicts, Kolloid Zeitschr. , 217 (1940).

[15] W. Bragg, Proc. Phys. Soc. , 105 (1940).

[16] C. Smith, *Private communication,* .

[17] J. Burke and D. Turnbull, *Recrystallization and grain growth,* Progress in Metal Physics , 220 (1952).

[18] *Surface tension Young-Laplace,* https://simple.wikipedia.org/wiki/Surface_tension, 29-07-2015.

[19] K. Aust and B. Chalmers, Proc. roy. Soc. , 210 (1950).

[20] C. Dunn, F. Daniels, and M. Bolton, Trans. Amer. Inst. min. (metall.) Engrs , 338 (1950).

[21] E. A. Lazar, *The Evolution of Cellular Structures via Curvature Flow*, Ph.D. thesis, Princeton University (2011).

[22] M. Gage and R. Hamilton, *The heat equation shrinking convex plane curves,* .

[23] M. Grayson, *The heat equation shrinks embedded plane curves to round points,* .

[24] M. Hillert, *On the theory of normal and abnormal grain growth,* Acta Metallurgica , 227 (1965).

[25] J. Sethian, *Theory, Algorithms, and Applications of Level Set Methods for propogating Interfaces,* //math.berkeley.edu/~sethian/2006/Papers/sethian.actanumerica.1995.pdf (1995), 29-06-2015.

[26] J. Sethian", *"level set explained by sethian",* math.berkeley.edu/~sethian/2006/Semiconductors/ieee_level_set_explain.html, 29-06-2015.

[27] S. Kim and D. I. Kim, *Computer simulations of two-dimensional and three-dimensional ideal grain growth,* .

[28] K. Janssens, *An introductory review of cellular automata modeling of moving grain boundaries in polycrystalline materials,* Mathematics and Computers in Simulation , 1361 (2010).

[29] M. M. Mul, *Modeling the austenite ferrite transformation by cellular automaton, improving interface stability,* Ph.D. thesis, TU Delft (2014).

[30] W. W. Mullins, *Two-Dimensional Motion of Idealized Grain Boundaries,* Journal of Applied Physics , 900 (1956).

[31] T. Tamaki, *Two-dimensional Grain Growth Simulation by Local Curvature Multi-vertex Model,* , 31 (2013).

[32] R. Fullman, *Metal interfaces,* .

[33] K. Fuchizaki, T. Kusaba, and K. Kawasaki, *Computer modelling of three-dimensional cellular pattern growth,* Philosophical Magazine Part B , 333 (1995).

[34] Y. B. J. Lepinoux, D.Weygand, *A vertex dynamics simulation of grain growth in two dimensions,* Philosophical Magazine B , 329 (1998).

[35] A. S. J. van Kan and F. Vermolen, *Numerical methods in scientific computing,* VSSD (2005).

[36] K. Kawasaki, T. Nagai, and K. Nakashima, *Vertex models for two-dimensional grain growth,* Philosophical Magazine Part B , 399 (1989).