A highly efficient 3D level-set grain growth algorithm tailored for ccNUMA architecture

C Mießen¹, N Velinov¹, G Gottstein¹ and L A Barrales-Mora¹,²

¹ Institute of Physical Metallurgy and Metal Physics, RWTH Aachen University, D-52056 Aachen, Germany
² School of Mechanical Engineering, Georgia Tech Lorraine, 2 Rue Marconi, F-57070 Metz, France

E-mail: miessen@imm.rwth-aachen.de

Received 20 January 2017, revised 14 August 2017
Accepted for publication 16 August 2017
Published 6 October 2017

Abstract
A highly efficient simulation model for 2D and 3D grain growth was developed based on the level-set method. The model introduces modern computational concepts to achieve excellent performance on parallel computer architectures. Strong scalability was measured on cache-coherent non-uniform memory access (ccNUMA) architectures. To achieve this, the proposed approach considers the application of local level-set functions at the grain level. Ideal and non-ideal grain growth was simulated in 3D with the objective to study the evolution of statistical representative volume elements in polycrystals. In addition, microstructure evolution in an anisotropic magnetic material affected by an external magnetic field was simulated.

Keywords: parallelization, level-set, large-scale simulation, grain growth, magnetically driven grain boundary motion

(Some figures may appear in colour only in the online journal)

1. Introduction
The evolution of the microstructure during the synthesis, processing and operation of materials has been an important research topic in materials science for decades. Among the different processes that cause modifications to the microstructure, grain growth has recently been the focus of interest because of its relevance in nanocrystalline materials. In order to elucidate this physical phenomenon more precisely, researchers have employed statistical analysis on results of computer simulations. Since the first related numerical algorithm was introduced in the 1980s and 90s, e.g. [1–5], available computational power has grown by
orders of magnitude. Despite this fact, microstructure evolution simulation seems to have benefited less from this development. Most algorithms still require prohibitively large amounts of time when simulating representative volume elements (RVE) of microstructure realizations, which are usually required to produce valid statistical results [6, 7]. Additionally, due to massive computational costs, most of these algorithms are not able to rigorously incorporate the effects of physical phenomena like inclination dependent grain boundary (GB) properties, properties of triple lines and quadruple junctions or energy densities.

In order to meet their demand for computational power, researchers have turned to computer clusters. As these resources are expensive to purchase and maintain they are shared between a large number of research groups. This results in limited availability and prioritization of these resources. For these reasons, it is very important to develop algorithms which are efficient and fast, but can still resolve the physical phenomena with comprehensive complexity and high accuracy.

In the specific case of grain growth, several approaches have been introduced in the past. They can be grouped into volume discretized methods, such as phase field models [8–11] or level-set (LS) models [12–25], explicit methods like vertex models [26–32], and probabilistic approaches, such as Monte Carlo Potts models [33–37]. In order to reduce the run-time of these simulations, some of these models were developed to be executed on modern computer architectures [7, 19, 24, 38–41]. Most algorithms utilize a domain decomposition strategy, which results in explicit communication paths to exchange information across sub-domain walls usually utilizing a message passing interface (MPI). To resolve the original problem without any statistical correction, communication between computational processes is inevitable. The costs depend, however, substantially on the implementation.

By contrast, the algorithm proposed in [22] divides the microstructure into grains, and duplicates their GBs to create generally independent objects. In this contribution, we present a new implementation of this LS algorithm for the simulation of grain growth in two and three dimensions. We showed in a previous publication [22] that our model is capable of incorporating a variety of factors influencing grain growth, such as anisotropic GB energies, GB mobilities, and in particular finite junction mobilities in 2D. In this contribution, we discuss the implementation details of the algorithm, and its efficiency for parallel computing applications. An explicit domain decomposition was deliberately avoided and replaced by a dissection along the GBs to facilitate parallelization. As the algorithm implements the grain growth phenomena as competing shrinkage/growth of adjacent grains, data has to be exchanged across GBs to reconstruct the network. The majority of computational effort is localized to individual objects, the grains, to predict their surface evolution. By these simple modifications, it is possible to render the algorithm parallelizable to a great extent as discussed in subsequent sections. Our implementation utilizes the OpenMP application programming interface for parallelization. Although OpenMP targets shared memory architectures, we were able to tailor our implementation to a specific cache-coherent non-uniform memory access (ccNUMA) architecture.

More importantly, though, handling grains as individual objects enables us to track the evolution of individual grains and their neighborhood, thus their topological paths [38], which gives the opportunity to study the dependencies of the different mesoscopic features of GBs and their effect on the local topological transitions. The introduced application (GraGLeS\(^3\)) is provided as open source code.

3 https://github.com/GraGLeS.
2. Simulation model

2.1. The level-set approach to grain growth

The LS method is a mathematical framework used to define surfaces and track their evolution in time. Since the method was introduced by Osher and Sethian [42], it has been applied to a large number of physical phenomena involving the motion of surfaces in any dimension. This method is capable of coupling the motion of multiple phases separated by sharp interfaces. The enclosing surface of a single region is represented by its own LS function. Thus, an interface is modeled by two congruent surfaces of adjacent regions. In the specific case of grain growth, the LS method implements the physical phenomena as a competing shrinkage of adjacent grains and automatically solves any topological transition and the extinction of entire phases or grains.

2.1.1. Level-set framework. An LS function is a real-valued function defined on the domain \( \Omega \). The surrounding GB (\( \Gamma \)) of an individual grain at time \( t \) can be described by means of the LS function \( \phi(t, x) \):

\[
\Gamma := \{ x \in \Omega \mid \phi(0, x) = 0 \} \subset \Omega.
\]

(1)

This set describes the level zero isosurface of the LS function. Instead of computing the motion of an explicit parametrization of the GBs, the method tracks the evolution of the LS function \( \phi \) and its particular isosurface with level zero (the GB), namely \( \Gamma \).

There are infinitely many possible functions that fulfill the condition in equation (1). By convention, if \( x \) is inside the grain, \( \phi \) will be positive, otherwise negative. To obtain a unique and regular definition of \( \phi(t, x) \), another constraint is introduced, namely:

\[
|\nabla \phi| = 1, \forall x \in \Omega.
\]

(2)

An LS function \( \phi(t, x) \) that fulfills equation (2) is called a signed-distance function (SDF) and will be denoted by \( d \) for distinction. This restriction is important to simplify the motion equation of the LS function. In addition, geometric quantities can be directly extracted from the LS function, such as the unit normal vector and the mean curvature, and can be further simplified by means of \( d \) (equation (2)):

\[
n = \frac{\nabla \phi}{|\nabla \phi|} = \nabla d
\]

(3a)

\[
\kappa = \nabla \cdot n = \nabla \cdot \left( \frac{\nabla \phi}{|\nabla \phi|} \right) = \Delta d.
\]

(3b)

Due to the sign convention defined above, the unit normal (equation (3a)) always points to the interior of the considered grain. Construction of the total derivative of \( \Gamma(t) \) (equation (1)) leads to the evolution equation for the LS function [22]:

\[
\frac{\partial \phi}{\partial t} - v_n |\nabla \phi| = 0.
\]

(4)

Here, \( v_n \) denotes the velocity in the normal direction of the isosurface at any point \( x \in \Omega \). To illustrate this equation, imagine moving all isosurfaces simultaneously into their normal directions with a velocity according to the local gradient. For our purposes, we assign the motion of an initial surface to the motion of the representing zero LS of the SDF function with time.
2.1.2. Modeling grain growth. Polycrystalline materials are composed of nearly perfect but differently oriented single crystals. The interfaces separating those crystals or grains are considered as defects within the overall (poly)crystal and are called GBs. Their properties depend on their own atomic structure and have proven themselves to be extremely complex [43–47]. Annealing induces a process of GB migration called grain growth. It reduces the amount of stored energy in the system by elimination of the GBs. This happens when the grain structure of the polycrystal reorganizes and eliminates the defects separating homogeneous regions of atoms. The name is ambiguous, as some grains shrink or completely disappear but the remaining grains grow in a statistical average. This process causes a reduction of the GB area, which is the defected area and thus, the source of the free energy.

In a mesoscopic approach to this problem, like the LS method, the complexity of the underlying system is described on a larger length scale than the atomic one [48]. On this scale, boundary migration during grain growth can be abstracted as driven by the energy density and therefore, the curvature of the boundary. In turn, the velocity of the migrating boundary can be calculated as:

\[ v(t, x(t)) = \mu(p_B - \gamma \kappa(x)) \text{.} \]  

Here \( \mu(\theta) \), \( \gamma(\theta) \), \( \kappa \) an \( p_B \) denote, respectively, the GB mobility, the GB energy, the local curvature of the GB and any additional driving force such as bulk energy densities resulting from anisotropic magnetic susceptibilities (equation (17)) or orientation dependent stored elastic energy densities. The entire evolution of a polycrystal can be understood as an energy minimizing process, where equation (5) gives the gradient descent of the 2D energy functional [12, 13, 36, 37]:

\[ E = \sum_{i<j} \text{length}(\Gamma_{ij}) \gamma(\theta)_{ij} + \sum_i \text{area}(\Sigma_i) \text{.} \]  

where \( \Sigma_i \) denotes grain \( i \). In the LS approach, a GB \( \Gamma_{ij} \) separating two adjacent grains is interpreted as a part of the surface of each individual phase or grain. Since the global gradient flow can be expressed as a composition of local flows, each grain can be considered as an individual entity [21]. Each entity is represented by one LS function and its isosurface. Owing to this disintegration of the microstructure, voids and overlaps can appear during their independent evolution. A constraint recomposes the microstructure in a procedure generally referred to as a predictor–corrector scheme.

Essentially, this gradient flow describes a certain energy dissipation of a space tessellation. The crucial observation is that the topology of the tessellation changes with time. The topological transitions proceed by the annihilation of GBs as well as the extinction of complete regions (grains). This leads to certain challenges in the modeling process, e.g. vertex models need explicit instructions to solve such transformations. By contrast, the LS method has no need for such computational instructions as it automatically handles these transitions with an additional corrector step (cf algorithm 1 described below). This step balances the predicted motion of free surfaces of individual grains and resolves the equilibrium state at the junctions.

2.1.3. Algorithm. There has been a considerable amount of research in the development of the LS method for grain growth [17–25]. This contribution focuses on an efficient implementation of this method and demonstrates some surprising results regarding the subsequent gain in computational performance. To begin with, the numerical scheme with special emphasis on the implementation demands will be introduced.
In this scheme, a polycrystal encompasses the region $\Omega : [0, 1]^D \subset \mathbb{R}^D$, where $D$ denotes the dimension, in particular 2 or 3. We assume that there is a tessellation $S$ of $\Omega$ that represents a polycrystal. Each sub-region $S_i$ is described initially as a polytope and represents a grain of this polycrystal. For a numerical description, each element $S_i \in S$ owns an LS function $\phi_i(x, t)$ (equation (1)) on a discrete domain $\Omega_i \subset \mathbb{N}^D$. In order to have a specific amount of points within each grain on average, the discretization of the computational domain $\Omega$ is first defined. Based on the overall dimension of the represented grain, $\Omega_i$ is extended by a certain offset of additional grid points to encompass the entire GB of the corresponding grain by means of its LS function $\phi_i(x, t)$ (figure 1(a)).

The LS function $\phi_i$ is initialized to fulfill equation (2), thus the scheme represents each grain by its SDF ($d_i$). The corresponding grid $\Omega_i$ must also be size-adaptive to accommodate the change in the size of the grain. Having defined $d_i \forall i \in I$, the GB migration of a $D$-dim ($D \in \{2, 3\}$) polycrystal can be computed with algorithm 1 proposed in [17–19, 22] and shown below.

**Algorithm 1.** LS algorithm for grain growth

1: **Initialization:** Create all $\Omega_i$ for all subset $S_i$ of the tessellation $S$
2: while $t < T_{\text{end}}$ do
3: **Predictor step:** Compute the motion of individual grains
4: **Corrector step:** Remove voids and overlaps
5: **Re-initialization:** Compute distance functions to new interface positions
6: end while

Figure 1. (a) A single 3D grain is described by its own LS function. The GB is rendered as an isosurface with level zero. Here the local domain $\Omega_i$ has a size of $47 \times 47 \times 47$ grid points, whereas there is the need to keep at least seven additional grid points into each spatial direction. The distance function was only evaluated in a tubular region enclosing the GB ($|\phi_i| \leq \epsilon$), outside it was restricted to $\epsilon = -7 \cdot h = -0.078$, where $h$ is the grid spacing. The palette indicates the value of the SDF. (b) The same isosurface is colored by its physical GB character $\mu \cdot \gamma$. Triple junctions, GBs and even quadruple junctions can be identified by our scheme and hence can be associated with different physical properties.

In this scheme, a polycrystal encompasses the region $\Omega : [0, 1]^D \subset \mathbb{R}^D$, where $D$ denotes the dimension, in particular 2 or 3. We assume that there is a tessellation $S$ of $\Omega$ that represents a polycrystal. Each sub-region $S_i$ is described initially as a polytope and represents a grain of this polycrystal. For a numerical description, each element $S_i \in S$ owns an LS function $\phi_i(x, t)$ (equation (1)) on a discrete domain $\Omega_i \subset \mathbb{N}^D$. In order to have a specific amount of points within each grain on average, the discretization of the computational domain $\Omega$ is first defined. Based on the overall dimension of the represented grain, $\Omega_i$ is extended by a certain offset of additional grid points to encompass the entire GB of the corresponding grain by means of its LS function $\phi_i(x, t)$ (figure 1(a)).
2.1.4. Predictor step. In order to simplify the driving force originating from the GB curvature, the anisotropic velocity of the GB \((\mu \gamma = 1)\) is normalized to simplify equation (5) as:

\[
\frac{\partial d}{\partial t} - \Delta d = 0.
\]  

(7)

The contribution of the bulk energy will be considered later. This differential equation describes the heat flow with \(d_t\) corresponding to the temperature distribution. The Green’s function \(G(t, x)\) is known for this problem and given by the Gaussian kernel [49]:

\[
G(t, x) = \frac{1}{(4\pi t)^{1/2}} \exp \left( -\frac{|x|^2}{4t} \right).
\]  

(8)

Convolving the distance function with the Green’s function for this problem [49] gives a solution to initial value problem \(d(0, \cdot) = g_0\):

\[
d(t, x) = G(t, x) * d(0, x).
\]  

(9)

Utilizing this approach, motion by mean curvature of an isosurface can be computed in any dimension. Note that this convolution can be efficiently solved in Fourier space. To extend this approach to a more complex driving force, which is weighted by a local anisotropy of the GB \((w)\), we developed a correction procedure in [22]. For this, we assumed that the energy and the mobility at any point on the GB is known and that each grid point has adopted this feature from the closest point on the GB. The subsequent correction evaluates two states of the LS function, an SDF \((d^k)\) and a convolved LS function \((d^p)\):

\[
\begin{align*}
\nu_n &= -(d_k - d^p_k)/(\Delta t) = l_i \quad &d^p_k(x, t) &= d_k(x, t) + (\nu w \Delta t) + (\mu \Delta E \Delta t).
\end{align*}
\]  

(10)

The quantities affecting the curvature are normalized to the interval \([0, 1]\) using the given maximal physical values for mobility \(\mu_{\text{max}}\) and energy \(\gamma_{\text{max}}\) for normalization. In the simplest case, the bulk energy is homogeneously distributed for each grain, whereas the driving force stems from the gradient descent of both potentials across the GB. These energies (magnetic or stored elastic) can also be given in physical units but have to be normalized to account for the dimension of the sample and the physical problem at hand.

To derive the correct normalization of the bulk energy, we consider a spherical grain with radius \(r\). \(L(D)\) denotes the dimension of the domain in one spatial direction represented by the unit interval \([0, 1]\). We replaced all variables in equation (5) by their corresponding normalized equivalents. Note that an asterisk * will always indicate that a variable is contained in the normalized space, whereas a normal notation will assign the physical equivalent. This spherical grain is assumed to be in an equilibrium of forces resulting in \(\nu_n = 0\) (equation (5)):

\[
\kappa = \frac{-2}{r} = \frac{-2}{r^* : L(D)} = \frac{p_B}{\gamma}.
\]  

(11)

where \(r^* \in (0, 1)\). Thus, we deduce a normalization coefficient \(c_B\) for a driving force \(p_B\) originating from a bulk energy given in physical units:

\[
c_B = \frac{L(D)}{\gamma_{\text{max}}}.
\]  

(12)

In addition, it is very important to connect the real (physical) time increment \(\Delta t\) with the internal time increment \(\Delta t^*\). Again, we replace the physical quantities by their corresponding
ones in the normalized space to deduce the connection:

\[
\Delta t = \Delta r \frac{\Delta r^* \cdot L(D)}{v_n \cdot \frac{1}{L(D)} \cdot \mu_{\text{max}} \cdot \gamma_{\text{max}}}
\]

\[
= v_n^* \cdot \frac{\Delta t^* \cdot L(D)^2}{\mu_{\text{max}} \cdot \gamma_{\text{max}}}
\]

This relation is very advantageous as it enables comparing simulation results with experimental findings and analytical solutions of certain growth phenomena.

2.1.5. Corrector step. Voids and overlaps appear during the predictor procedure because the grains are modeled as independent objects, whose evolution leads naturally to these features. The corrector step redistributes the critical grid points in those regions among all competing grains to reach an equilibrium state. This procedure ensures that the computational domain stays entirely covered by disjunct phases. More importantly though, this step automatically handles any topological transition and the elimination of entire grains. In practice, the comparison step is done by matching the predicted LS functions \(d^g_k, k \in \{1 \ldots M\}\) of adjacent grains:

\[
d^g_k(x, t + \Delta t) = \frac{1}{2}(d^g_k(x, t + \Delta t) - \max_{j \neq k} d^g_j(x, t + \Delta t)).
\]

2.1.6. Re-initialization. Finally, a re-initialization of all LS functions \(d^g_k\) to the corresponding zero LSs is needed to maintain numerical stability. Solutions to this problem have been provided in terms of equidistant grids in [50–53]. The re-initialization of the SDF faces two problems. First, the position of the interface has to remain invariant. Second, the gradient of the re-initialized LS function must fulfill \(|\nabla \phi| = 1\) to ensure a correct evaluation of the mean curvature in the subsequent predictor step. The formal complexity of the available methods differs substantially. By investing higher computational costs (\(\sim 40\) times more), higher order accuracy of the resulting signed-distance is possible [50–52]. This re-initialization approach originates from the reformulation of the eikonal equation as an evolution equation introduced first by Sussman et al [52]:

\[
\phi_t + S(\phi_{t=0})(|\nabla \phi| - 1) = 0.
\]

Here, \(\tau\) is an artificial time step size. A major drawback of this approach is that the zero LS is considerably displaced and the deviation could even increase with the number of iterations used to solve the ordinary differential equation (ODE; equation (16)). Russo and Smereka [51] improved the stability of the method in such a way that the displacement of the zero LS is independent of the number of iterations used to solve equation (16). This was achieved by using only information from one side of the zero LS in the discretization of the ODE.

The reformulation by Hartmann et al [50] is a modification of the partial differential equation introduced by Sussman et al [52] and, in particular, improves the second-order accurate modification proposed by Russo and Smereka [51]. The latter authors proposed two formulations of the re-initialization problem, namely, to explicitly minimize the displacement of the zero LS within the re-initialization and to solve the overdetermined problem to preserve the interface locally within the re-initialization. Thus, they proposed a direct update of the distance values next to the zero LS. Subsequently, the re-initialization starts from the side of the interface providing the most points along this segment of the isosurface.
By contrast, a fast-sweeping algorithm as proposed in [53] reconstructs the corresponding signed-distance values only by first order accuracy but still minimizes the displacement of the zero LS. Starting from a fixed set of distance values next to the interface the distance information is spread into all spatial directions incrementing with the grid spacing \( \Delta h \). Thus, the resulting distance values are given as taxicab\(^4\) distances to the closest grid point next to the interface. The distance calculation of these reference points is exact.

Such algorithms are usually appraised depending on the accuracy of the reconstructed SDF function related to a well-discretized but complex shape. Alternatively, they are also valued for their computational performance in applications where high accuracy is not required such as in video games. During grain growth, grain shapes resemble in most cases polyhedra with curved faces and therefore sufficient accuracy of well-discretized grains can be easily obtained. The challenge are rather vanishing grains, which are always poorly discretized and thus the evaluation of their curvature is hardly possible. Unfortunately, these grains can influence substantially the growth kinetics of the network as their released volume will be occupied by their adjacent grains and thus, the shrinkage regulates the growth. The effect is somehow inverse to what was found for finite triple junction mobilities, which affect the global kinetics by dragging the shrinkage of the smallest grains close to elimination [22, 31, 32].

Note that for certain purposes, LS functions are sometimes defined on unstructured meshes [15, 16, 23, 24, 54]. In such situations, the redistancing problem can become more expensive regarding the numerical effort. In order to couple the LS model with finite element methods the usage of unstructured meshes is indispensable because it narrows the computational effort to the regions of interest. Recent developments [55] have tackled the numerical difficulties originating from unstructured grids and reduced the resulting numerical costs.

Owing to the characteristic of the model described in this contribution, unstructured grids do not provide any benefit. For this reason, we focused on regular grids. A comparison between the aforementioned approaches for regular grids is given in figure 3(a) and is discussed in section 6.1.

3. Novel numerical approaches

3.1. Data structure

In this section, the implementation details which were crucial for the resulting code performance will be presented. They influence both sequential and parallel execution times. The programming language C++ was selected as it provides user defined memory allocation and allows us to structure the program in accordance with the physical model, using the object-oriented paradigm. Additional libraries like FFTW [56], Intel MKL [57] and EIGEN [58] were used for some mathematical algorithms.

The input data for the simulation is a tessellation \( S \) (as described in section 2.1.3). For each element \( S_i \subseteq S \), a grain object is created. The grain contains physical information (orientation, neighborhood, etc.) and represents the grain in LS space. Hence, the main purpose of the grain object is to keep track of the distance values \( B_i := \{ d_i(x) \mid x \in \Omega_i \} \)

\(^4\) The taxicab metric or the Manhattan norm, proposed by Hermann Minkowski in nineteenth-century Germany, replaces the Euclidean metric, the distance between two points, by the sum of the absolute differences of their Cartesian coordinates. The name traces back to the grid layout of most streets on the island of Manhattan, where the shortest path a car could take between two intersections has a length equal to the intersections’ distance in taxicab geometry.
As every step of the algorithm transforms the function \( d_i \), in some way, we keep two buffers \( B^1_i \) and \( B^2_i \) in each object. One is used as input to the current computation step whereas the other serves as output. Consequently, references to those buffers are switched back and forth after each step of algorithm 1; this is usually called double buffering. These buffers have to accommodate size changes of the corresponding grain. To accomplish this, when a buffer grows it is increased with more memory than requested, in order to accommodate future growth. This heuristic, combined with the grid-coarsening procedure (cf section 3.2 below), reduces the relocation of objects in memory substantially.

During the **corrector step**, information has to be exchanged among competing grains to redistribute the space along their boundaries and junctions (section 2.1.5). As grains are hosted by surrounding boxes, those boxes give a suitable instrument to compute a pre-selection of possible neighbors to a certain grain. In the initial time step, an R-tree is used to find all overlapping boxes as input for the comparison routine [59]. Subsequently, the local neighborhood is updated by the evaluation of the points of contact to preselected neighbor candidates. Memory fetches are only performed on the resulting box intersections \( (x \in \Omega_i \cap \Omega_j) \) with the already identified neighbors. Furthermore, for each point in \( \Omega_i \), we locally collect information about the nearest neighboring grain to that point (grain \( j \mid \max_{x \in \Omega_i} d_i(x) \)) in a structure called *ID-field*. For more precise neighbor statistics, the *ID-field* (which is part of the grain object) is evaluated next to the isosurface of level zero, i.e. the GB.

All other subroutines of the algorithm are local (in terms of memory fetches) to the volume of the local grid \( \Omega_i \), hence to the grain object itself. As we are aiming at efficient parallel computations, the structure of the grain objects allows very flexible task scheduling since sets of grains can be directly assigned to computational threads. A production job will always use all 128 threads available, where each thread will get a similar workload. For example, an initial structure composed of 500,000 grains would be spread to approximately 3,900 grains per thread.

### 3.2. Sequential optimizations

In section 2.1.4, it was mentioned that each point in \( \Omega_i \) needs to adopt the physical properties of the nearest point on the GB. Mathematically, this mapping was an easy problem that, nonetheless, required a tremendous numerical effort. The SDF was inappropriate to evaluate this connection as it provided solely scalar information in the form of distance values. Therefore, the position of the GB had to be computed first using a marching squares/cubes algorithm depending on the dimension to obtain an explicit representation [60]. Hence, the closest snippet of that linear representation to each grid point had to be identified. Unfortunately, there were, to our knowledge, no algorithms in 2D or in 3D that could solve the point to polygon and the point to triangulation problem in sub-linear complexity. In both cases, the complexity is dominated by the cardinality of the set of linear segments or triangles \( O(E) \), where \( E \) is the number of edges of the polygon.

In order to reduce the computational effort, we opted to cluster the linear representations of the grains according to the number of neighboring grains next to each piece of the linearization using the procedure described in section 3.1. Utilizing this information about the GB character itself, it was possible to distinguish between segments forming a GB, a triple line, a quadruple junction or even high order junctions. This concept and the utilization of the *ID-field* led to a substantial reduction of the complexity of the search for the next closest segment on the GB. Usually, a triangulation of the entire GB is composed of a couple of thousand triangles in 3D, while in 2D the isosurface is composed of only hundreds of linear
segments. By clustering those snippets reusing the ID-field, the search is constrained to a fraction of the original set. This subset contains just those snippets associated to one GB and the adjacent junctions. Additionally, the expensive evaluation was restricted to the set of points enclosed by a narrow tube around the GB.

As a further sequential optimization of the code, a grid-coarsening routine was applied, which kept the spatial resolution of the grains constant with time. Thus, the grid spacing grew inversely proportional to the population so that the average discretization per grain remained constant. The projection of each SDF to a coarser grid was performed by a tri-linear interpolation scheme. The improvement was substantial as it not only reduced the memory consumption with time but also allowed increasing the time step simultaneously, with proportionality $1/N^2$.

3.3. Parallelization

Due to the grain-resolved data structure, each step of algorithm 1 could be executed independently for each grain. Remote memory was accessed only during the corrector step as each grain needs to refer to the $\Omega_j$s of all adjacent grains to correct its own distance function. Hence, synchronization of grains was needed after the end of each subroutine, which meant that all grains must have completed the previous step of the algorithm with regard to the current iteration. The parallelization idea was very natural, as we just distributed grains among the nodes contributing to the simulation (figure 2).

Our implementation targeted a certain computer architecture, a machine operated by BULL, consisting of up to 128 cores coupled to one big shared memory system: therefore, four physical boards owning four processors (Intel) were connected with BULL’s Coherent Switch (BCS) chips. Thus, not only the interconnection between those four boards imposed a non-uniform memory access (NUMA) topology but also each board consisted of four NUMA nodes connected via another QPI (Quick Path Interface). This is important to note as memory

---

**Figure 2.** The scatter of grains to computational threads is illustrated by coloring grains to indicate their affiliation to a certain thread ID. Here, the four different colors refer to the four physical boards, which host 32 threads each. Furthermore, each color gradation maps to a fixed thread ID. We implemented two complementary concepts, the first (figure 2(a)) aiming just for a homogeneous workload distribution and the second (figure 2(b)) also for the reduction of remote memory access during the corrector step (section 2.1.5).
fetches from remote locations will lead to different latencies on the two level NUMAness. This issue will be further discussed in section 3.4.

Essential to the parallel performance of our implementation was that each thread utilized only memory local to its corresponding core. We achieved this by employing a two step process. First, each thread was bound to a specific core using Linux’s NUMA API\(^5\) and affinity control API\(^6\). Second, instead of using the built-in heap manager of the compiler, we used Jason Evans malloc (jemalloc)\(^7\) [61]. This allocator ensured that each thread got a memory pool, in which all allocations and deallocations were executed. The allocation policy combined with the thread binding and the first touch policy\(^8\) of the Linux kernel allowed optimal memory placement for the simulation. We stress that these optimizations are not specifically bound to our algorithm and could in fact be used in any simulation that would benefit from thread-local memory allocation and consumption.

The parallel performance of the predictor step (algorithm 1) needed certain consideration. Here, the distance function of each grain \(d_i\) had to be transformed to Fourier space, where the appropriate kernel was applied as described in section 2.1.3. To accomplish this, we tested two libraries—FFTW [56] and Intel MKL\(^9\). Both libraries require a certain pre-computation effort, in the form of a computational plan that is created before the actual transformation can take place. Due to the implementation of FFTW, this precomputation could not be executed in parallel, and thus, had to be always serialized. This serialization hindered the parallelization of the predictor step substantially. Therefore, we opted to use Intel MKL in the subsequent simulations, where the computation plan could be done in parallel and no serialization was required. Furthermore, we could reuse the plans whenever the size of the bounding box had not changed with time. To save allocation effort for these out-of-place transformations each thread operated its own buffer reused by all grains in the related workload.

3.4. Task distribution

First, a random distribution of grains to threads was implemented to achieve a homogeneous workload balance on all participating computational threads (figure 2(a)). For all subroutines, with the exception of the corrector step, this strategy might be optimal as computations were limited to local memory. By contrast, the corrector step contains a lot of remote memory access as it recomposes the network by comparing grains owned by different threads. Due to the described NUMA architecture of the hardware, these remote fetches (from another board) are very expensive—up to 10-times slower than local fetches (inter-thread communication on one board is not critical). The remote access could also be graphically identified by the amount of GBs separating differently colored grains, as we illustrated the scatter by coloring grains to indicate the affiliation to a thread ID and the corresponding board (figure 2). We will evaluate the effect of this naively designed patchwork structure (figure 2(a)) compared against the spatial scheduling (figure 2(b)) in the next chapter.

---

\(^5\) http://linux.die.net/man/3/numa.
\(^6\) http://man7.org/linux/man-pages/man2/sched_setaffinity.2.html.
\(^7\) Jemalloc is a general purpose malloc(3) implementation that emphasizes fragmentation avoidance and scalable concurrency support.
\(^8\) The first touch policy states that each virtual page is allocated in physical memory only when touched for the first time.
\(^9\) https://software.intel.com/en-us/intel-mkl.
4. Simulation setups

4.1. Model validation

To validate the simulation program, a simple case study was constructed that provides an analytical solution for the relevant driving forces. For a spherical grain embedded in a continuum, the theoretical course of the radius evolution was compared to the results of the simulation. Starting from a reasonable discretization of the spherical grain, this setup was utilized to evaluate the differences of the two competing re-initialization schemes introduced in section 2.1.6, the most advanced by Hartmann et al [50] and the fastest suggested by Zhao [53] to recompute the SDF to the new position of the interface after each time step.

In addition to the curvature (also referred to as capillary) driving force, we considered a bulk energy originating from a magnetic field [62–66]. During heat treatment of certain metallic materials with anisotropic magnetic susceptibility, such as bismuth, titanium or zirconium, the microstructure evolution can be affected by an external magnetic field as shown in several experiments [63–65]. The magnetic field can affect GB migration [67] as an additional driving pressure on the GBs which emerges due to anisotropic magnetic susceptibilities causing the magnetic energy density to become dependent on the grain orientations with respect to the field direction:

\[ E_m = \frac{1}{2} \mu_0 H^2 \chi \]  

(17)

where \( \mu_0, \chi, H \) denote the magnetic permeability of the vacuum, the magnetic susceptibility and the magnetic field strength, respectively. According to elementary crystallography, the magnetic susceptibility of a single crystal can be written as \( \chi = (\chi_{||} + \Delta \chi \cos^2 \theta) \), where \( \Delta \chi \) is the susceptibility difference of grains with their \( c \)-axis parallel \( \chi_{||} \) and perpendicular \( \chi_{\perp} \) to the magnetic field direction. Hence, the magnetic energy density of a grain reaches its maximum when the \( c \)-axis of the grain becomes parallel to the magnetic field direction whereas it becomes minimal when its \( c \)-axis is perpendicular to it. The additional driving pressure results from the difference in the volumetric energy between two adjacent grains \( i, j \):

\[ p_m = \frac{1}{2} \mu_0 \Delta \chi H^2 (\cos^2 \theta_i - \cos^2 \theta_j). \]  

(18)

Then, the simulation setup consisted of a spherical grain with a radius of 50 \( \mu m \) and an orientation of \( (\phi_1 = 180^\circ, \Phi = 35^\circ, \phi_2 = 0^\circ) \) in Bunge–Euler notation. This grain was embedded in a continuum with an orientation \( (\phi_1 = 0^\circ, \Phi = 35^\circ, \phi_2 = 0^\circ) \). In terms of the magnetic susceptibility we took the value for titanium from the literature [63] \( \Delta \chi = 1.18 \times 10^{-5} \). The magnetic field direction was chosen to be \( (0^\circ, -32, 8^\circ, 46.93^\circ) \). Thus the \( c \)-axis of the continuum or matrix grain was exactly perpendicular to the field, whereas the spherical grain’s \( c \)-axis was approximately parallel to maximize the magnetic driving pressure. For various magnetic field strengths the paths of the spherical grain were tracked and compared against the corresponding analytic solution.

4.2. Performance analysis

For the performance analysis of our implementation, a benchmark problem was defined, which was qualified to evaluate the performance in sequential as well as in parallel execution. For these studies, we chose the re-initialization scheme which was found to be the most beneficial. A network composed of 100,000 grains in 2D and 10,000 grains in 3D defined a rather realistic problem size which could still be computed without parallelization while the
workload was sufficient to study any performance issues in parallel. The grains were initially discretized by 20 points on average per spatial direction and the networks were evolved for 20 integration time steps.

To investigate the memory consumption and the domain superimposition, we increased the problem size in 3D to 100,000 grains and simulated the coarsening until only 100 grains remained in the system. Such a polycrystal consumed nearly 300 GB peak memory in the first time steps. It is thus sufficient in size to trace a long term evolution of the heap memory.

4.3. Grain growth with consideration of bulk energy

As an application of our grain growth model, the case of growth affected by a magnetic field was studied. There has been a considerable amount of work in this field providing numerous experimental results for comparison [62–65]. Simulations were also performed utilizing a vertex model [30, 68]. These simulations were capable of reproducing the essential experimental findings. It is stressed, however, that these simulations were performed only in 2D owing to the numerical complexity of implementing a vertex model in 3D [66, 69].

A polycrystal with 500,000 grains in 3D was initialized utilizing a total number of 959^3 grid points which equals a memory peak of about 1.2 TB. The grain orientations were randomly sampled from a reference set of orientations measured in titanium using EBSD [62]. The resulting texture was bimodal and very similar to the experimental one (figure 8(d)). Hence, all grains had orientations close to \( O_1 = (180^\circ, 35^\circ, 0^\circ) \) or \( O_2 = (0^\circ, 35^\circ, 0^\circ) \). The rolling direction (RD) of the sample was set perpendicular to the magnetic field whereas the transverse direction (TD) was tilted 32° with respect to the field, which corresponded to a magnetic vector \((0^\circ, \cos(32^\circ), \sin(32^\circ))\). Thus, the grains either had a c-axis approximately perpendicular or parallel to the applied magnetic field. The magnetic field generated a magnetic flux density of 17 Tesla, which corresponded to the experiment carried out in [62].

The grains initially had a size of \( d = 20\mu m \) in diameter. Depending on crystal orientation, the grains carried an additional magnetic energy in the range of \( E_m \in [0, 1351.23] \frac{J}{m^3} \). We assumed a GB energy of 0.3 J/m² and a high angle GB mobility of \( 3 \cdot 10^{-11} m^4 / J s \) for titanium. For low misorientations the GB energy depended on the disorientation angle \( \theta \) between the adjacent grains following the Read–Shockley model [70], whereas for \( \theta > 15^\circ \) the GB was considered to be a high angle GB associated with a constant energy. In terms of the GB mobility, we distinguished generally between low and high angle GBs but defined a steep ascent between 8–10 degrees misorientation:

\[
\mu_{GB} = 1.0 - \left( 0.99 \cdot \exp \left( -5 \cdot \left( \frac{\theta}{15^\circ} \right)^3 \right) \right)
\]

The simulations were performed until only 1000 grains were left in the microstructure.

5. Results

5.1. Model validation results

To begin with, we will report the findings for the spherical grain. As a mapping between physical and internal units was introduced in section 2.1.4, we were able to evaluate all simulation results in physical units. In figure 3(a), the shrinking grain was tracked utilizing two different re-initialization algorithms. No difference was observed until 1,500 s annealing. Reaching approximately a diameter of 7–8 grid points, an acceleration of the shrinkage was
observed in the simulation, where the scheme of Hartmann et al \cite{50} was used. In turn, the
evolution of the kinetics was slightly delayed for the extremely fast method of Zhao \cite{53}.
Still, a better agreement in terms of the arrival time was found using the latter method.

For this reason, subsequent simulations were performed exclusively utilizing Zhao’s
approach. Afterward, we validated the effect of a secondary driving force originating from an
external magnetic field in figure 3(b). A broad range of magnetic field strengths was applied
to capture cases where even the magnetic driving force exceeded the capillary pressure and
forced the grain to grow. Excellent accuracy was found for all realizations of magnetic field
forces (figure 3(b)).

5.2. Performance

The program was designed to be executed on a BULL SMP-S (BCS) machine of the RWTH
Aachen computer cluster. These machines are composed of 128 cores grouped into two
NUMA levels. The processor type is an Intel Xeon X7550 with 2.00 GHz clock rate (eight
cores on one processor chip). Four cores on each chip share their L3 cache, whereas the L2
and L1 cache memories are individually owned by each core. Four chips (each eight cores)
are connected via a QPI and form the first NUMA level. Four clusters of these 32 core
aggregates are connected via BCS to build the entire ccNUMA architecture. The BCS con-
nection is very similar to the QPI but slower. The RAM is limited to 128 GB on BULL SMP-
S, whereas BULL SMP-XL offers 2 TB. These machines were also used to perform all of the
simulations reported in this study.

5.2.1. Sequential performance. In a serial run of our benchmark example in 2D, the
computational costs of each subroutine of algorithm 1 were measured. In the initial version
the anisotropy correction, introduced in detail in sections 2.1.4 and 3.2, consumed up to 87%
of the computational costs. As this part of the algorithm was targeted by our sequential
optimizations (section 3.2), the resulting benefit is evaluated in figure 4. By just optimizing
the anisotropy correction, the run-time of the benchmark example was reduced to one third in
2D and one tenth in 3D.
5.2.2. Parallel performance. It is useful to clearly distinguish between savings made by the optimization of the sequential algorithm (section 5.2.1) and the speedup gained by parallelization. The execution of our benchmark case in 2D was repeated using 1, 16, 32, 64 and 128 cores and measured the run-time for the subroutines to obtain a detailed overview of the scaling. We will directly report the results using the two step procedure introduced in section 3.3, bounding threads to cores and utilizing the jemalloc allocator (as replacement of the built-in heap manager of the compiler) to assure that the first touch policy always guarantees a local memory allocation.

For all subroutines, with the exception of the comparison step, a monotonic trend in the speedup was observed. In the best case (that of the anisotropy correction), a super-linear acceleration of 145 times using only 128 threads was achieved, whereas the worst scaling was found for the MKL library functions resulting in a speedup of only 46. The other subroutines scaled approximately linearly with different slopes. The average behavior, resulting in the total scaling and denoted in figure 6 as Main 2D iterative, still reached a speedup of 107 compared to the sequential run-time. Since the execution was limited to one NUMA node, i.e. up to 32 cores (figure 5), linear scaling was also observed for the corrector step. Consulting more cores, a massive delay was observed for this procedure, the only subroutine fetching memory remotely. The execution on 64 cores took double the time compared to the one with 32 cores.

By changing the scheduling routine, we aimed to improve the scaling of the corrector procedure. As all other subroutines accessed only local memory, their scaling behavior was insensitive to the scheduling procedure. For this reason, remote memory access via the BCS paths was suspected to cause the poor parallel performance. From figure 5, we concluded that inter-thread communication during the comparison was not an issue as long as all threads were hosted by the same NUMA node (maximum of 32 threads). Therefore, we formed teams

Figure 4. Shares of the subroutines during a sequential execution of the benchmark configuration computing 20 integration steps with time for a polycrystal composed by 100 000 grains (2D). As the convolution correction on the left turned out to be the alone-standing bottleneck of our scheme, we optimized that routine and reduced the formal complexity by a factor of four without losing accuracy. The area of the circles correspond to the execution time of the main function (3h versus 1h). The sequential run-time was improved by a factor of 2.95 from left to right.
of 32 threads, hosted on the same node, to execute the workload of a spatially bounded sub-region of the entire domain (figure 2(b)). We decomposed the network into four sub-regions; equal to the number of NUMA nodes. As we explicitly pinned threads to cores, forcing the threads \(0, \ldots, 31\) to run on the first NUMA node, \(32, \ldots, 63\) on the second node and so on, we introduced a spatial scatter of threads. Aiming to reduce the remote memory access via the BCS we introduced a mapping between the physically connected regions and the teams of threads, i.e. the NUMA nodes. As the utilized allocation policy guaranteed thread-local memory allocation, this memory placement strategy hosted those grains close together in the memory which were also physically nearby (figure 2(b)). Figure 6 illustrates the related improvements made in the critical part of the algorithm. As a result, the corrector procedure was substantially improved reaching almost linear scaling. For the 3D benchmark case, we observed a very similar trend. The savings made by the scheduling were even more important as the data exchange rate increased exponentially owing to the higher dimensionality.

For the anisotropy correction, super-linear scaling behavior moved its computational cost closer to the mean (figure 5). Super-linear speedup resulted from the cache effect originating from different memory hierarchies of the BCS SMP-S machine because not only the number of processors changed but also the size of accumulated caches from different processors and the number of memory access units. Our benchmark simulation in 2D consumed about 13 GB RAM memory, which exceeded the 8 GB memory owned by the operating processor. For this reason, the remaining memory was allocated non-locally to the processor, which caused higher latencies. Hence, distributing the memory in the heap reduced the access time dramatically, which caused the extra speedup in addition to that from the actual computation.

The total execution time for the ultimate solution in 2D was 32 s after computing 20 integration steps of the benchmark sample containing 100,000 grains and 270 s for the 3D case containing 10,000 grains (table 1).

5.2.3. Memory consumption. For an analysis of the memory usage, a bigger network was initialized—a 3D polycrystal composed of 100,000 grains. The simulation was run in parallel.

Figure 5. Strong scaling was investigated for a fixed size problem of 100,000 grains in 2D. The main subroutines are presented individually, whereas the ideal speedup is given as reference (dashed line). While the total speedup is rather good, we achieved an acceleration of 107.5x utilizing 128 threads. The biggest obstacle is caused by the corrector routine. Here we observed a break down in speedup moving to the second NUMA level of our hardware architecture.
utilizing 128 threads on a BSC SMP-XL node. For all computations with memory requirements, expected to be larger than 128 GB, these jobs were scheduled to an identically constructed but taller system, a BULL SMP-XL node, with a capacity of 2 TB RAM. However, the evolution of the memory consumption did not depend on the number of threads used. The results had to be classified as a sequential optimization to the model but it was hardly possible to run a production job of suitable sample size on a single core.

The memory consumption and the domain superimposition with time (figure 7) were studied in order to compare our implementation to global level-set (GLS) approaches from the literature [19, 23]. The domain superimposition averages the duplicates of each grid point used to store all grains in local containers. Thus, this frequency is directly comparable to the number of GLS functions in former approaches. To determine this frequency, we added up the number of grid points used in all our sub-domains and divided it by the size of the entire domain. Figure 7 documents the substantial saving in memory consumption in 3D. The memory peak of the simulation was found at around 280 GB RAM.

The memory consumption first increased and then dropped somehow exponentially. Surprisingly, the domain superimposition did not follow exactly the same trend but was

| Subroutine                  | 2D (100 000 grains) | 3D (10 000 grains) |
|-----------------------------|---------------------|---------------------|
| Anisotropy                  | 18 s                | 165 s               |
| Redistancing                | 2.9 s               | 21.9 s              |
| Marching squares/cubes      | 0.6 s               | 13.8 s              |
| Helper                      | 0.9 s               | 3.7 s               |
| FFT                         | 4 s                 | 15.5 s              |
| Comparison (iterative/spatial) | 10.5 s/2.7 s      | 116 s/48 s          |

Figure 6. Spatial scheduling of grains (figure 2(b)) improved the execution time of the comparison routine to average behavior. Communication in between computational threads, performed as remote memory access no longer delayed the scaling of the algorithm. The 3D benchmark study also utilizing spatial scheduling showed a slightly better speedup than the 2D version. Due to the parallelization the total execution times of the two benchmark cases decreased from 1h:03 min to ~29 s in 2D respectively, from 8h:25 min to <5 min in 3D.
We found that the volume covered by all sub-domains did not exceed 12 times the size of the entire domain in 3D.

5.3. Large-scale simulation—grain growth affected by an external magnetic field

We modeled a microstructure comparable to the one observed in experiments in titanium polycrystals to investigate the microstructure and texture evolution during conventional and magnetic annealing of this material (figures 8(d) to (f)). Grains were clustered by their affiliation to the preferential texture components observed in experiments for rolled titanium [62]. Hence, we could differentiate sub-populations of grains and track their evolution with time. The evolution of the mean radius of the texture components O1 and O2, respectively which were disfavored or favored because of their different energy density according to equation (17), was plotted in figure 9.

For comparison, the same polycrystal was simulated without the influence of a magnetic field and its mean radius evolution was added to figure 9(a). Starting from a population of 500,000 grains, we were able to study a long annealing time of about 1100 s. Due to the magnetic field the volume of grains near orientation O1, whose c-axis was approximately parallel to the field, decreased continuously (figures 8(d) to (f)) in the presence of the magnetic field (17 T). Grains of this disfavored orientation were continuously eliminated as is obvious from figure 9(b). Caused by this growth selection, the final population was dominated by grains with a c-axis orientation nearly perpendicular to the applied field (figures 8(f) and 9(b)). By contrast, the reference simulation (same initial polycrystal without the presence of a magnetic field) showed no texture modification in favor of a certain component (figure 9(a)). Simultaneous to the elimination of disfavored grains, the kinetics of the process slowed down. As the sub-population of favored grains increased and started occupying the whole sample volume, their size gain slowed down compared to the one observed in the reference simulation (figure 9(a)).
Figure 8. (a)–(c) Time series (at 10 s, 55 s, 500 s annealing) of a polycrystal with initially 500,000 grains on a grid of 9593 points was annealed under the influence of an external magnetic field until only 1000 grains remained in the sample. Grains were colored by their magnetic energy $E_m$ [J/m$^3$]. The microstructure gradually eliminated grains with a high magnetic susceptibility, colored in red, to reduce the amount of free energy in the system. (d)–(f) Corresponding [0002] pole figures. During annealing the disfavored texture component disappeared nearly completely.

Figure 9. (a) Texture evolution of the polycrystal as an evolution of the mean grain size of the preference orientations O1 and O2. Simulation of the same polycrystal without a field indicated a strong growth selection for grains bigger than 20 μm in radius. (b) Fraction of favored and disfavored grains over time during magnetic annealing.
6. Discussion

6.1. Numerical accuracy

Regarding the numerical accuracy, the effect of two competing re-initialization schemes, namely, those by Hartmann et al [50] and by Zhao [53] (section 2.1.6) were evaluated. Deviations from the analytical expectation for the shrinking spherical (figure 3(a)) grain were caused by the accumulated numerical error of the entire scheme and not just the re-initialization. The convolution itself had a numerical error as it evaluated the curvature of the isosurface depending on the resolution of the grain [21]. The convolution of an SDF with a Gaussian kernel acts as a smoothing of the entire SDF function. If the number of grid points separating two facing GBs becomes sufficiently small, these GBs will start attracting each other and thus, the shrinkage of the grain will be accelerated. This expected behavior was observed for the scheme of Hartmann et al [50].

It was very surprising that the simpler re-initialization scheme by Zhao resulted in better agreement concerning the arrival time of the shrinking grain. The reason for this is the annihilation of numerical errors during the evaluation of the curvature and the re-initialization of the SDF. As the taxicab distance (the distance estimation in the scheme by Zhao is very similar) is always larger than the Euclidean distance, the gradient of the SDF was set to be slightly steeper than the exact solution. The subsequent convolution of a slightly distorted SDF function was certainly affected. Here, the steeper gradient of the SDF opposed the attraction of facing GBs as the resolution of the interior of the grain became very low.

Since this phenomenon is characteristic of vanishing grains, we decided to proceed with the latter scheme proposed by Zhao [53]. However, we want to emphasize that the arrival time is decisive for the overall kinetics of grain growth as the released volume of a shrinking grain will be occupied subsequently by the adjacent grains. Thus, the entire network kinetics would be strongly affected by this circumstance. Note that we do not claim that the accuracy of Zhao’s scheme is better than the one obtained with the Hartmann et al method but it is more favorable to proceed with the Zhao scheme for three simple reasons: (i) the accuracy of the entire algorithm is higher in the case of vanishing grains, (ii) the arrival time matches best and (iii) the computational effort is much lower.

6.2. Performance improvement

The purpose of this contribution was to increase the productivity of grain growth simulations and related phenomena considering the capabilities of modern computer architecture. We opted for a grain-related mathematical description, as provided by the LS method, and subsequently chose an object-related implementation utilizing local LS functions on local grids. This implementation strategy, described in section 3, was the key factor in the performance of the resulting simulation software (figures 5 and 6). The major difference between our approach and other similar models was that our data structure allowed for grain-resolved parallelism in the sense that each grain could be processed independently as an individual entity. Other approaches [17, 18, 20, 23–25] stored grains in GLS functions and subsequently processed these clusters simultaneously. We believe that our approach provides the following advantages:

- Small sub-grids allow for precisely controlled memory layout and placement.
- Grain-resolved parallelism enables for a flexible workload distribution among all computing threads.
- Operations on local LS functions have a much higher cache efficiency compared to GLS.
• Reduced memory consumption; the frequency of domain superimposition was limited in 3D by only a factor of 12 compared to 64 GLS functions as reported in [18] and two versus 32 in 2D.
• Regarding each grain as a single object allows tracking the evolution of certain grains and their topological paths individually, which gives the opportunity to study the dependencies of the different macroscopic features of grains and their effect on the local topological transitions [38].

A grain-related workload distribution among threads was not successful at once. Without a solitary thread binding policy, threads would migrate over participating cores substantially limiting the parallel performance. The migration causes expensive re-caching of memory. Still, a thread binding policy does not handle memory placement directly. A first touch policy combined with the built-in heap manager of the Intel compiler is supposed to allocate memory locally to the operating thread but gives no guarantee in practice. A small chunk of memory can be by chance placed everywhere in the address space resulting in poorer parallel performance. To cope with this problem we turned to jemalloc, which clusters the heap into arenas associated to a set of threads—four by default. Each arena is operated by its own heap manager and thus an allocation local to the thread was guaranteed. This coherence of task scheduling, thread binding policy and memory allocation is critical for any shared memory application. Hence, this major result regarding parallel performance (section 3.3) is a portable and beneficial concept for all applications capitalizing from assured thread-local memory allocation.

For illustration purposes, runtimes of recent simulation studies using GLS are compared in table 2. The former authors did not specify precisely the utilized computer architecture and thus, a direct comparison is therefore not completely fair. Nevertheless, Elsey et al [19] used Intel Nehalem i7 processors, which have a quad-core chip with 2.8 GHz to 3.33 GHz CPU frequency. By contrast, Scholtes et al [23] only specified the utilized architecture by the number of Intel Xeon CPUs with 1.2 GHz CPU frequency, most likely also a quad-core chip. It remains therefore uncertain which chip was exactly used and more importantly how the chips were linked in both cases.

Despite the uncertainty regarding the used computer architecture, the computational times of the different implementation strategies, GLS versus the local LS, are significantly different. As a consequence, the problem sizes that can be tackled can vary strongly. Due to the achieved parallel efficiency and the reduction of memory utilization, the solution presented in this contribution enabled computing the evolution of 500,000 grains in 3D down to 1000 grains within a day considering anisotropic GB properties and an additional driving force. Even though it was not possible to utilize the model on an equivalent powerful computer architecture, as we were limited to 16 CPUs (eight cores each; sections 3.3 and 5.2.1), we could gain orders of magnitude in run-time solving an equal size grain growth problem compared to former LS approaches (table 2).

6.3. Grain growth

Owing to the new implementation and parallelization strategy, the computational performance of the LS method was significantly improved. This achievement enabled analyzing more comprehensively grain growth studies by incorporating more accurate statistics. This is important regarding the problem size that defines an RVE [73]. Certainly, this size depends on the features to be observed.
Table 2. Review of the computational performance of recent large-scale simulation studies. For comparability, the setups were resampled using the same discretization as in the literature. All simulations were run on a BULL SMP-XL machine composed of 16 CPUs with eight cores each as specified in sections 3.3 and 5.2.1.

| Method                                      | 2D (50 000 grains) / isotropic growth | 3D (5000 grains) / isotropic growth | 3D (64 000 grains) / anisotropic growth | 3D (240 000 grains) / isotropic growth |
|---------------------------------------------|--------------------------------------|------------------------------------|----------------------------------------|---------------------------------------|
| GLS FE by Scholtes et al [23]              | 80 h on 32 CPUs                      | 5 h on 60 CPUs                     | 4-5 days on 48 CPUs                    |                                       |
| GLS convolution by Elsey et al [19]        |                                       |                                    |                                        |                                       |
| This contribution                          | < 1 min on 16 CPUs                   | < 12 min on 16 CPUs                | 1:40 h on 16 CPUs                      | 10.5 h on 128 cores (16 CPUs)         |
| *Open Phase* [71, 72]                      |                                       |                                    |                                        | 200 h on 512 cores                    |
Grain growth of isotropic GB networks, i.e. equal GB energies and mobilities, are expected to arrive in a steady state, referred to as a self-similar state. The steady state is characterized by means of a constant relative grain size distribution. The duration of the transient is usually unknown but depends on the initial GSD and its difference to the equilibrium steady state distribution \[74, 75\].

In order to characterize the stationary GSD, a convenient formulation of the GSD function in 3D was given by Rios et al:

\[
p(z, \nu) = \frac{z^{2}H^{2}}{(\rho^{2}z^{2} - \nu^{2})^{3/2}} \cdot \exp \left[ - \frac{Hv}{\sqrt{4\nu - \nu^{2}}} \left( \arctan \left( \frac{2\rho z - \nu}{\sqrt{4\nu - \nu^{2}}} \right) \right) \right. \\
+ \left. \arctan \left( \frac{\nu}{\sqrt{4\nu - \nu^{2}}} \right) \right] \\
\]  

(20)

where \( z = \frac{R}{\langle R \rangle}, \rho = \frac{\langle R \rangle}{R_{c}} = \frac{\langle R \rangle}{R_{c}} = \text{const.} \) and \( H = 3 \) \[76, 77\]. In the limit \( \nu \to 4 \) with \( \rho = 0.8 \), equation (20) yields the Hillert distribution (figure 10(a)) \[78\]. \( \rho \) characterizes the ratio between average grain size \( \langle R \rangle \) and the critical grain size \( R_{c} \). The latter is mostly used for theoretical considerations and determines the variance of the distribution:

\[
\text{Var} \left( \frac{R}{\langle R \rangle} \right) = \frac{1}{\langle R \rangle^{2}} (E(R^{2}) - (E(R))^{2}) \approx \frac{1}{\langle R \rangle^{2}} (\langle R^{2} \rangle - \langle R \rangle^{2}) = \frac{1}{\rho} - 1. \]

(21)

Furthermore, it holds for the expected value \( E(z) = \left( \frac{R}{\langle R \rangle} \right) = 1. \) The remaining parameter \( \nu \) affects the skewness of the probability function (equation (20)). Since the variance is fully defined by \( \rho \) (as \( E(z) = 1 \)), it is possible to determine the steady state from an analysis of \( \rho \) alone. A large-scale simulation was therefore conducted to ensure a sufficient number of remaining grains till the network arrives in a steady state.

The expected self-similar state was not reached before the initial population was thinned out to a fifth (\( \sim 100,000 \) grains) starting also from a Voronoi tessellation (figures 10(a) and

![Figure 10](image_url)

Figure 10. (a) Relative grain size distribution converges to a steady state characterized by equation (20) for \( \rho = 0.863 \). (b) During the transient towards the stationary relative grain size distribution, 80% of the initial population was eliminated. The convergence of the parameter \( \rho \) indicates a steady state of the grain size distribution. A value of \( \rho = 0.889 \), which corresponds to Hillert’s GSD, does not characterize the steady state.
We stress that the Hillert distribution was not found to match with the observed population, neither during the transient phase nor in the steady state. The steady state could be characterized by the following parametrization of the GSD given in equation \((20)\):

\[
\langle R \rangle^n - \langle R_0 \rangle^n = k \cdot t
\]

For isotropic grain growth conditions, values of the growth exponent usually spread in the range of 0.45–0.55 [17, 23, 83, 84]. Such deviations might originate from a lack of statistics, i.e. considering too few grains. In order to investigate the dependency on problem size, three isotropic grain growth simulations were conducted starting from 5000, 50 000 and 500 000 grains with an initial grain size \(1 \mu m\) each using periodic boundary conditions. From figure 10(b), it is already known that the transient phase takes at least \(\sim 40\) s of computational annealing. In the smaller simulations, it would not be possible to determine this transient exactly. The root equation (equation \((23)\)) was therefore fitted cutting of the initial data points from \(0 < t < 87\) s. At 87 s annealing, the average grain size had increased to 2.5 \(\mu m\) and the population was certainly in a self-similar regime. The resulting values for the best fit of the kinetic exponent are plotted in figure 11(a). It is evident that the simulation with initially 5000 and 50 000 grains did not provide enough data. Here, the resulting exponent was very sensitive on the left cutoff range and therefore the results experienced significant deviations. Former simulation results were performed in systems containing at most 30 000 grains [17, 23, 83–85]. Based on these simulation studies different exponents have been determined to describe the evolution of the average grain size by means of the Burke–Turnbull equation. The dependency on the sample size (figure 11(a)) clearly indicates that this variation originates from sampling too small populations.

For a steady state population, decreasing from 500 000 grains to 1000 grains, an exponent of \(n = 0.507 \pm 0.002\) was observed in figure 11(a) using periodic boundary conditions. For the largest simulation the resulting kinetic exponent was observed to be practically insensitive to the cutoff time. Despite the big variation of the exponent for the smaller simulation setups it is interesting to investigate the validity of their growth prediction. In order to extrapolate their kinetics, the exponents found for 40 s annealing were utilized. Interestingly, this data turned out to be sufficient to approximate the long term kinetics of the large-scale simulation with less than 10% deviation. Still, the quality of this extrapolation depends on the precise knowledge of the transient phase towards the self-similar state and therefore the deviation of the extrapolation will differ stronger using a slightly different data range.
6.4. Grain growth with consideration of additional driving forces

In this contribution, it was opted to simulate magnetically driven GB motion because the magnitude of the driving force can be calculated clearly from the physical conditions without relying on any further assumptions. In our case, the magnetic field direction was chosen in such a way that half of the population was disfavored whereas the growth of the other half was promoted. The divergence of the volume evolution of both texture components was fast for grains with a radius between $10^{-40}$ mm–until the share of O1 (approximately parallel to the field) was heavily thinned out. Afterward, grains of O2 lost their growth advantage as their neighborhood transformed and hence, these grains shared only low angle GBs with a substantially lower mobility. Finally, more than 99% of the sample volume was occupied by grains of orientation O2. Without magnetic field, both components O1 and O2 developed in a similar fashion. The entire kinetic of the polycrystal influenced by the magnetic field dropped behind the reference one due to the loss of high angle, respectively, fast GBs. Although the magnetic field was estimated to be an order of magnitude smaller than the capillary driving force $r/2g$, its impact was strong. However, this estimation of the capillary force is just a rough approximation and therefore it can hardly explain the growth selection in the early phase of the growth. In a polycrystal the GB curvature is much smaller than the one of an equally sized spherical shape, thus the impact of the magnetic field on the kinetic was reasonable. It is worth mentioning that the results agree very well with 2D vertex model simulations of the same phenomenon [68]. This indicates in principle that the dimensionality does not substantially affect the results of the simulations as both kinetics and texture evolution were predicted by both models in excellent agreement with experiments.

7. Conclusions

An efficient algorithm based on the level-set approach for the simulation of 2D and 3D grain growth has been presented. By optimizing the data structure and optimizing the sequential and parallel parts of the algorithm, we could speed up our simulations by three orders of magnitude. The most important innovation was the developed data structure, which divided
the polycrystal into macroscopic objects—the grains. A spatial distribution of grains to computational threads nearly eliminated remote access across the shared memory space. Thus, strong scaling of our application up to 128 threads on our test architecture at the RWTH Aachen computer cluster was achieved. The achieved parallelization of the model allows the simulation of very large grain ensembles. With simple modifications to the algorithm, it is possible to tailor the model to more traditional computer architectures such as distributed memory systems. This would allow simulating even larger microstructures.

A real time scaling and rescaling of physical units was introduced allowing for easy comparability to experiments. In an application of the computational scheme, we simulated the evolution of a 3D polycrystal with 500,000 grains affected by a magnetic field. The findings on texture evolution and kinetics were compared to theoretical expectations, experimental results and previous simulations showing in all cases an excellent agreement. The implemented model showed excellent computational performance. More importantly though, its greatest advantage is that it allows incorporating transparently different physical effects. The consideration of additional energy densities permits the simulation of not only grain growth but also recrystallization related phenomena.

Acknowledgments

The authors gratefully acknowledge the financial support from the Deutsche Forschungsgemeinschaft (DFG) within the ‘Reinhart Koselleck-Project’ (GO 335/44-1) as well as the FZJülich and the RWTH Aachen University for granting computing time within the frame of the JARAHPC project no. 0076.

ORCID iDs

C Mießen @ https://orcid.org/0000-0002-3507-4355
L A Barrales-Mora @ https://orcid.org/0000-0002-7998-6815

References

[1] Anderson M, Srolovitz D, Grest G and Sahni P 1984 Computer simulation of grain growth-I kinetics Acta Metall. 32 783–91
[2] Srolovitz D, Anderson M, Sahni P and Grest G 1984 Computer simulation of grain growth-II grain size distribution, topology, and local dynamics Acta Metall. 32 793–802
[3] Grest G S, Anderson M P and Srolovitz D J 1988 Domain-growth kinetics for the Q-state Potts model in two and three dimensions Phys. Rev. B 38 4752–60
[4] Hesselbarth H and Gobel I 1991 Simulation of recrystallization by cellular automata Acta Metall. Mater 39 2135–43
[5] Brakke K A 1992 The surface evolver Exp. Math 1 141–65
[6] Kühbach M, Gottstein G and Barrales-Mora L-A 2016 A statistical ensemble cellular automaton microstructure model for primary recrystallization Acta Mater. 107 366–76
[7] Kühbach M, Barrales-Mora L A and Gottstein G 2014 A massively parallel cellular automaton for the simulation of recrystallization Modelling Simul. Mater. Sci. Eng. 22 075016
[8] Chen L-Q 2002 Phase-field models for microstructure evolution Annu. Rev. Mater. Res. 32 113–40
[9] Steinbach I, Pezzolla F, Nestler B, Seeleberg M, Prieler R, Schmitz G J and Rezende J L L 1996 A phase field concept for multiphase systems Physica D 94 135–47
[10] Kazaryan A, Wang Y, Dregia S A and Patton B R 2002 Grain growth in anisotropic systems: comparison of effects of energy and mobility Acta Mater. 50 2491–502
[11] Kazaryan A, Patton B R, Dregia S A and Wang Y 2002 On the theory of grain growth in systems with anisotropic boundary mobility Acta Mater. 50 499–510
[12] Zhao H-K, Chan T, Merriman B and Osher S 1996 A variational level set approach to multiphase motion J. Comput. Phys. 127 179–95
[13] Zhao H-K, Osher S, Chan T and Merriman B 1996 A variational level set approach to multiphase motion Journal of Computational Physics 127 179–95
[14] Merriman B, Bence J K and Osher S J 1994 Motion of multiple junctions: a level set approach J. Comput. Phys. 112 334–63
[15] Bernacki M, Chastel Y, Coupez T and Logé R 2008 Level set framework for the numerical modelling of primary recrystallization in polycrystalline materials Scr. Mater. 58 1129–32
[16] Bernacki M, Digonnet H, Resk H, Coupez T, Logé R, Cesar de Sa J M and Santos A D 2007 Development of numerical tools for the multiscale modelling of recrystallization in metals, based on a digital material framework AIP Conf. Proc. 908 375–80
[17] Elsey M, Esedoglu S and Smerka P 2009 Diffusion generated motion for grain growth in two and three dimensions J. Comput. Phys. 228 8015–33
[18] Elsey M, Esedoglu S and Smerka P 2010 Large-scale simulation of normal grain growth via diffusion-generated motion Proc. R. Soc. A 467 381–401
[19] Elsey M, Esedoglu S and Smerka P 2013 Simulations of anisotropic grain growth: efficient algorithms and misorientation distributions Acta Mater. 61 2033–43
[20] Elsey M and Esedoglu S 2014 Fast and accurate redistancing by directional optimization SIAM J. Sci. Comput. 36 A219–31
[21] Esedoglu S, Ruuth S and Tsai R 2010 Diffusion generated motion using signed distance functions J. Comput. Phys. 229 1017–42
[22] Mießen C, Liesenjohann M, Barrales-Mora L A, Shvindlerman L S and Gottstein G 2015 An advanced level set approach to grain growth accounting for grain boundary anisotropy and finite triple junction mobility Acta Mater. 99 39–48
[23] Scholtes B, Shakoor M, Settefrati A, Bouchard P-O, Bozzolo N and Bernacki M 2015 New finite element developments for the full field modeling of microstructural evolutions using the level-set method Comput. Mater. Sci. 109 388–98
[24] Scholtes B, Boulais-Sinou R, Settefrati A, Muoz D P, Poitrault I, Montouchet A, Bozzolo N and Bernacki M 2016 3D level set modeling of static recrystallization considering stored energy fields Comput. Mater. Sci. 122 57–71
[25] Hallberg H A modified level set approach to 2D modeling of dynamic recrystallization Modelling Simul. Mater. Sci. Eng. 21 085012
[26] Kawasaki K, Nagai T and Nakashima K 1989 Vertex models for two-dimensional grain growth Philos. Mag. B 60 399–421
[27] Weygand D, Bréchet Y and Lépinoux J 1998 A vertex dynamics simulation of grain growth in two dimensions Philos. Mag. B 78 329–52
[28] Barrales-Mora L A, Shvindlerman L S, Mohles V and Gottstein G 2007 The effect of grain boundary junctions on grain microstructure evolution: 3D vertex simulation Mater. Sci. Forum 558–559 1051–6
[29] Barrales-Mora L A 2008 2D and 3D grain growth modeling and simulation PhD Thesis Cuvieller Verlag, Göttingen
[30] Barrales Mora L A 2010 2D vertex modeling for the simulation of grain growth and related phenomena Math. Comput. Simul. 80 1411–27
[31] Barrales Mora L A, Mohles V, Shvindlerman L S and Gottstein G 2008 Effect of a finite quadruple junction mobility on grain microstructure evolution: theory and simulation Acta Mater. 56 1151–64
[32] Barrales-Mora L A, Gottstein G and Shvindlerman L S 2012 Effect of a finite boundary junction mobility on the growth rate of grains in two-dimensional polycrystals Acta Mater. 60 546–55
[33] Zöllner D and Streitenberger P 2006 Three-dimensional normal grain growth: Monte Carlo Potts model simulation and analytical mean field theory Scr. Mater. 54 1697–702
[34] Zöllner D 2011 A Potts model for junction limited grain growth Comput. Mater. Sci. 50 2712–9
[35] Zöllner D 2012 Grain microstructure evolution in two-dimensional polycrystals under limited junction mobility Scr. Mater. 67 41–4
[36] Srolovitz D, Grest G and Anderson M 1986 Computer simulation of recrystallization homogeneous nucleation and growth Acta Metall. 34 1833–45
[37] Srolovitz D, Grest G, Anderson M and Rollett A 1988 Computer simulation of recrystallization heterogeneous nucleation and growth Acta Metall. 36 2115–28
[38] Kühlbach M, Barrales-Mora L-A, Mielten C and Gottstein G Ultrafast analysis of individual grain behavior during grain growth by parallel computing 89 012031
[39] Suwa Y, Saito Y and Onodera H 2007 Three-dimensional phase field simulation of the effect of anisotropy in grain-boundary mobility on growth kinetics and morphology of grain structure Comput. Mater. Sci. 40 40–50
[40] Suwa Y, Saito Y and Onodera H 2008 Parallel computer simulation of three-dimensional grain growth using the multi-phase-field model Mater. Trans 49 704–9
[41] Nestler B 2005 A 3D parallel simulator for crystal growth and solidification in complex alloy systems J. Cryst. Growth 275 e273–8
[42] Osher S and Sethian J A 1988 Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations J. Comput. Phys. 79 12–49
[43] Brandenburg J-E, Barrales-Mora L and Molodov D 2014 On migration and faceting of low-angle grain boundaries: experimental and computational study Acta Mater. 77 294–309
[44] Brandenburg J-E, Barrales-Mora L, Molodov D and Gottstein G 2013 Effect of inclination dependence of grain boundary energy on the mobility of tilt and non-tilt low-angle grain boundaries Scr. Mater. 68 980–3
[45] Barrales-Mora L A and Molodov D A 2016 Capillary-driven shrinkage of grains with tilt and mixed boundaries studied by molecular dynamics Acta Mater. 120 179–88
[46] Molodov D A, Barrales-Mora L A and Brandenburg J-E 2015 Grain boundary motion and grain rotation in aluminum bicrystals: recent experiments and simulations IOP Conf. Series: Materials Science and Engineering 89 012008
[47] Barrales-Mora L, Molodov D A and Brandenburg J E 2016 Effect of grain boundary geometry on grain rotation during curvature-driven grain shrinkage Diffusion Foundations 9 73–81
[48] Hoffrogge P W and Barrales-Mora L A 2017 Grain-resolved kinetics and rotation during grain growth of nanocrystalline aluminium by molecular dynamics Comput. Mater. Sci. 128 207–22
[49] Evans L 1998 Partial Differential Equations (Berkeley, CA: American Mathematical Society)
[50] Hartmann D, Meinke M and Schröder W 2008 Differential equation based constrained reinitialization for level set methods J. Comput. Phys. 227 6823–43
[51] Russo G and Smereka P 2000 A remark on computing distance functions J. Comput. Phys. 163 51–67
[52] Sussman M, Smereka P and Osher S 1994 A level set approach for computing solutions to incompressible two-phase flow J. Comput. Phys. 114 146–59
[53] Zhao H 2004 A fast sweeping method for eikonal equations Math. Comput. 74 603–27
[54] Bernacki M, Resk H, Coupez T and Logé R E 2009 Finite element model of primary recrystallization in polycrystalline aggregates using a level set framework Modelling Simul. Mater. Sci. Eng. 17 064006
[55] Shakoor M, Scholtes B, Bouchard P-O and Bernacki M 2015 An efficient and parallel level set reinitialization method application to micromechanics and microstructural evolutions Appl. Math. Modelling 39 7291–302
[56] Frigo M and Johnson S G 2005 The design and implementation of FFTW3 Proc. IEEE 93 216–31
[57] Intel Math Kernel Library. Reference Manual, Intel Corporation, 2009
[58] Guennebaud G, Jacob B et al 2010 Eigen v3 http://eigen.tuxfamily.org
[59] Guttman A 1984 R-trees: A dynamic index structure for spatial searching SIGMOD Rec. 14 47–57
[60] Rajon D A and Bolch W E 2003 Marching cube algorithm: review and trilinear interpolation adaptation for image-based dosimetric models Comput. Med. Imaging Graph. 27 411–35
[61] Evans J 2006 A scalable concurrent malloc(3) implementation for freebsd
[62] Molodov D A, Bollmann C, Konijnenberg P J, Barrales-Mora L A and Mohles V 2007 Annealing texture and microstructure evolution in titanium during grain growth in an external magnetic field Mater. Trans 48 2800–8
[63] Molodov D and Sheikh-Ali A 2004 Effect of magnetic field on texture evolution in titanium Acta Mater. 52 4377–83
[64] Molodov D, Bollmann C and Gottstein G 2007 Impact of a magnetic field on the annealing behavior of cold rolled titanium Mater. Sci. Eng. A 467 71–7
[65] Molodov D and Bozzolo N 2010 Observations on the effect of a magnetic field on the annealing texture and microstructure evolution in zirconium Acta Mater. 58 3568–81
[66] Barrales-Mora L A 2010 2D vertex modeling for the simulation of grain growth and related phenomena Math. Comput. Simul. 80 1411–27
[67] Mullins W W 1956 Two-dimensional motion of idealized grain boundaries J. Appl. Phys. 27 900–4
[68] Barrales-Mora L, Mohles V, Konijnenberg P and Molodov D 2007 A novel implementation for the simulation of 2-D grain growth with consideration to external energetic fields Comput. Mater. Sci. 39 160–5
[69] Barrales-Mora L A, Mohles V, Gottstein G and Shvindlerman L S 2009 Network and vertex models for grain growth in: ASM Handbook: Fundamentals of Modeling for Metals Processing vol 22a (OH: ASM International) pp 266–90
[70] Read W T and Shockley W 1950 Dislocation models of crystal grain boundaries Phys. Rev. 78 275–89
[71] Kamachali R D 2017 Extension of large-scale phase-field simulations of grain growth Private communication
[72] Tegeler M, Shchyglo O, Kamachali R D, Monas A, Steinbach I and Sutmann G 2017 Parallel multiphase field simulations with openphase Comput. Phys. Commun. 215 173–87
[73] Kühbach M, Gottstein G and Barrales-Mora L 2016 A statistical ensemble cellular automaton microstructure model for primary recrystallization Acta Mater. 107 366–76
[74] Rivier N 1985 Statistical crystallography structure of random cellular networks Philos. Mag. B 52 795–819
[75] Rivier N and Lissowski A 1982 On the correlation between sizes and shapes of cells in epithelial mosaics J. Phys. A: Math. Gen. 15 L143
[76] Rios P, Dalpian T, Brando V, Castro J and Oliveira A 2006 Comparison of analytical grain size distributions with three-dimensional computer simulations and experimental data Scr. Mater. 54 1633–7
[77] Rios P and Glicksman M 2008 Polyhedral model for self-similar grain growth Acta Mater. 56 1165–71
[78] Hillert M 1965 On the theory of normal and abnormal grain growth Acta Metall. 13 227–38
[79] Kamachali R D, Abbondandolo A, Siburg K and Steinbach I 2015 Geometrical grounds of mean field solutions for normal grain growth Acta Mater. 90 252–8
[80] Kamachali R D and Steinbach I 2012 3D phase-field simulation of grain growth: topological analysis versus mean-field approximations Acta Mater. 60 2719–28
[81] Rios P R and Glicksman M E 2006 Topological theory of abnormal grain growth Acta Mater. 54 5313–21
[82] Burke J and Turnbull D 1952 Recrystallization and grain growth Prog. Metal Phys. 3 220–92
[83] Zöllner D and Streitenberger P 2006 Three-dimensional normal grain growth: Monte Carlo Potts model simulation and analytical mean field theory Scr. Mater. 54 1697–702
[84] III C K and Chen L-Q 2002 Computer simulation of 3D grain growth using a phase-field model Acta Mater. 50 3059–75
[85] Darvishi Kamachali R and Steinbach I 2012 3D phase-field simulation of grain growth: topological analysis versus mean-field approximations Acta Mater. 60 2719–28