



Parallel Correction for Hierarchical Re-Distancing Using the Fast Marching Method

Michael Quell¹(✉), Georgios Diamantopoulos¹, Andreas Hössinger²,
Siegfried Selberherr³, and Josef Weinbub¹

¹ Christian Doppler Laboratory for High Performance TCAD,
Institute for Microelectronics, TU Wien, Vienna, Austria

{quell,diamantopoulos,weinbub}@iue.tuwien.ac.at

² Silvaco Europe Ltd., Saint Ives, UK

andreas.hoessinger@silvaco.com

³ Institute for Microelectronics, TU Wien, Vienna, Austria

selberherr@iue.tuwien.ac.at

Abstract. Topography simulation is typically implemented with the level-set method which uses the level-set function to represent the interface. The signed-distance property, capturing the distances from the entire simulation domain towards the interface, has to be regularly restored during a simulation in order to update the distances relative to the evolution of the interface. The restoring process is called ‘re-distancing’ and the most established algorithm is the Fast Marching Method. For Cartesian grids, which are commonly used, high-performance applications require an adaptive resolution for geometric features, such as narrow trenches or sharp corners. Among the most important challenges is the need to utilize the solution in higher resolved regions to correct the solution in the embedding coarser regions. We present a parallelized bottom-up (i.e. from finest to coarsest resolution levels) correction technique for hierarchical re-distancing using the Fast Marching Method, which increases the accuracy of the discretized level-set function on the coarser grids. The coarser grids are corrected by interpolation on grid points covered by finer regions and a partial restart of the Fast Marching Method for the remaining grid points, thus minimizing the computational effort. This parallel correction step has been integrated into a recently developed parallel re-distancing algorithm, is implemented in C++ using OpenMP, and is evaluated for different geometries. The correction step significantly reduces the error in the signed-distance function, introducing a performance penalty of less than 10%.

1 Introduction

The level-set method [1] is used in topography simulations to track deforming volumes $\Omega \subset \mathbb{R}^d$. This can be achieved by implicitly representing the boundary of

the volume as the zero level-set of a level-set function $\phi(\mathbf{x}, t)$. The time evolution of Ω is given by

$$\frac{\partial \phi}{\partial t} - V(\mathbf{x}, t) \cdot \nabla \phi = 0, \quad (1)$$

with V being the deforming velocity of Ω . A recent review on the level-set method and its applications, such as computational fluid dynamics, minimal surfaces, and epitaxial growth, is given in [2].

The signed-distance property ($|\nabla \phi(\mathbf{x})| = 1$) of the level-set function has to be regularly restored, since solving the level-set Eq. (1) does not generally preserve it [3]. This restoration procedure, called *re-distancing*, modifies a given level-set function ϕ so that the signed-distance property is fulfilled without changing the position of the zero level-set [4, 5]. This is equivalent to solving the Eikonal equation

$$\begin{aligned} |\nabla \phi(\mathbf{x})| &= f(\mathbf{x}) & \mathbf{x} \in \mathbb{R}^n, \\ \phi(\mathbf{x}) &= g(\mathbf{x}) & \mathbf{x} \in \partial\Omega, \end{aligned} \quad (2)$$

for $f(\mathbf{x}) = 1$ and $g(\mathbf{x}) = 0$.

The Fast Marching Method (FMM) [1] is a widely used method to solve (2). The FMM is a one-pass algorithm using upwind differences. However, the use of a priority queue in the algorithm prohibits straightforward, efficient parallelization. This gave rise to other algorithms, most importantly the Fast Sweeping Method (FSM) [6] and the Fast Iterative Method (FIM) [7]. Both methods have been shown to be not competitive regarding accuracy due to their iterative nature [8]. Additionally, applications of the level-set method often employ narrow-band re-distancing¹ to reduce computational costs, which is not possible for the FSM [9]. Recently, a new approach using the Hopf-Lax formula, which is also iterative, has been developed showing a good parallel speed-up. However, comparisons to the FMM have yet to be made [9].

In another approach, a parallel version of the FMM via a distributed-memory domain decomposition approach has been developed, offering excellent accuracy [10]. Based on this approach a shared-memory parallelized FMM for multiple meshes within a hierarchical grid was developed [11], which was later extended to handle multiple meshes within a hierarchical grid [12, 13].

Hierarchical grids enable covering geometrically challenging regions (e.g., corners, narrow trenches) with higher resolution (or *finer*) meshes on a separate, higher resolution grid level [14]. This method, therefore, supports higher accuracy in particular regions of interest without the need to globally increase the mesh resolution, thereby balancing the number of mesh elements with the accuracy. This is also known as adaptive mesh refinement.

As another consequence and the main context of this work, the accuracy of the solution in coarser grid levels can be increased by incorporating the more accurate solutions from the relevant finer levels, thereby correcting the local solution. In this work, such a correction algorithm tailored to the level-set method,

¹ The distance is only computed up to a given threshold value around the zero level-set [1].

is introduced and directly integrated into a FMM based re-distancing step with manageable computational overhead.

This work is structured as follows. Section 2 provides necessary background on the FMM. Section 3 introduces details of the employed hierarchy of grids and the scheme for re-distancing this hierarchy. Section 4 provides the details of the correction step. Section 5 evaluates the presented correction step with respect to improved accuracy using two-dimensional test cases. Additionally, computational benchmarks are presented to assess the parallel performance using a three-dimensional test case inspired by simulating a typical fabrication process step of a microelectronic structure.

2 Fast Marching Method

The FMM is based on the idea behind Dijkstra's algorithm [15] for finding the shortest path in a graph. The FMM has two stages: initialization and marching. The initialization sets for all grid points the flag **Far** and their distance to infinity. Next, the flag **KnownFix** is assigned to the set of *given* grid points (i.e., points for which the distances are known in advance) and their pre-defined distances are assigned accordingly. Their neighboring grid points with flag **Far** are assigned the flag **Band** and the distance is calculated based on the upwind finite difference discretization

$$\left[\begin{array}{l} \max \left(D_{ijk}^{-x} \phi, -D_{ijk}^{+x} \phi, 0 \right)^2 + \\ \max \left(D_{ijk}^{-y} \phi, -D_{ijk}^{+y} \phi, 0 \right)^2 + \\ \max \left(D_{ijk}^{-z} \phi, -D_{ijk}^{+z} \phi, 0 \right)^2 \end{array} \right]^{\frac{1}{2}} = 1 \quad (3)$$

with D_{ijk}^- and D_{ijk}^+ as the first order backward and forward difference operators [1].

For the marching stage, the grid point with the smallest distance to the interface and marked **Band** is set to **Known**. Subsequently, all its neighbors' distances are recalculated. If the newly calculated distance for a neighbor is smaller than the current distance, the new distance is assigned and the neighbor is marked **Band**. This is repeated until all grid points are flagged **Known** or **KnownFix**. In case of narrow-band re-distancing, the marching procedure is stopped, if the grid point with the smallest distance to the interface and marked **Band** is beyond the narrow-band threshold value.

3 Hierarchical Re-Distancing

Critical geometrical features of the interface, such as narrow trenches and sharp corners, require high spatial resolution to be appropriately represented [2]. As previously indicated, a hierarchical grid enables capturing of regions of interest by introducing additional hierarchy levels, offering locally defined meshes with

increased resolution and thus avoiding a globally increased mesh resolution. In the following an overview of hierarchical grids and hierarchical re-distancing is given.

The here considered hierarchical grid consists of several levels. Level 1 is the coarsest grid. A subsequential Level N is refined by a factor \mathbf{n}_{N-1} compared to Level $N - 1$. Different resolutions for each spatial dimensions are possible. Every level holds a set of meshes, to represent the regions of interest which require a higher mesh resolution. Data is stored in a cell-centered manner.

We consider the following hierarchical grid properties: On Level 1 there is always only one mesh covering the whole domain. The meshes on the finer levels must conform to the following: (1) Meshes must not overlap, (2) a mesh must have exactly one parent mesh in the next coarser grid, (3) the meshes are aligned to the parent grid points, and (4) all the grid points in the parent mesh have neighboring grid points on the same refinement level. Figure 1 shows a two-dimensional example.

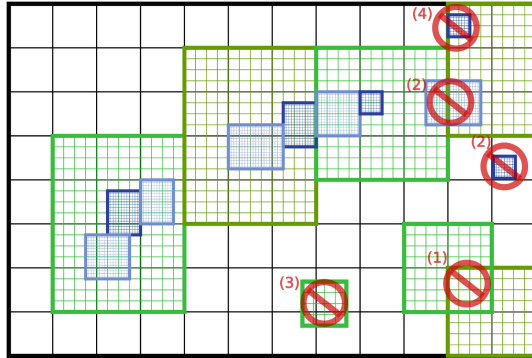


Fig. 1. Two-dimensional hierarchical grid with depth 3 and refinement factor 4 for all spatial directions and levels. Invalid configurations are highlighted in red with the number corresponding to the violated rules (1)–(4).

Hierarchical Re-Distancing

The re-distancing begins on Level 1, as this is the only mesh for which all the boundary conditions, i.e. the domain boundaries, are set. The FMM is initialized with the grid points set by the domain boundaries and the grid points which have a neighboring grid point with the opposite sign, since they are next to the interface and must not be modified. After the FMM finished processing a level, the boundaries of the meshes on the next finer level are set by interpolation and re-distancing is performed on it. This procedure is repeated for every level until the finest level is calculated.

The processing of a grid level has been parallelized by a coarse grained approach treating the individual meshes in parallel and enforcing an explicit data

exchange on shared mesh boundaries via ghost layers [13]. This is implemented using OpenMP² task directives, where each mesh is processed by a separate task.

4 Correction Step

The developed correction step is based on the previously described hierarchical re-distancing step (i.e., the flags and distances are set according to the re-distancing algorithm [13]). The bottom-up correction begins on the finest level. For notational coherence as the correction spans over two levels, meshes on the finer level will be called child meshes and the meshes on the next coarser level are called parent meshes. Algorithm 1 shows the pseudocode of the correction step algorithm.

Algorithm 1: Correction Step

```

// From finest to coarsest level
1 ChildLevel ← FinestLevel
2 ParentLevel ← FinestLevel.parentLevel()
3 while ChildLevel != CoarsestLevel do
    // Initialization step
4   for every Parent mesh in parallel do
5     for every Child mesh in parallel do
6       ChildList ← interpolateParentGridPoints()
7       mergeChildLists()
8       initializeFMMCorrection(ChildList)
9   Barrier
    // Marching step (identical to the original FMM)
10  for every Parent mesh in parallel do
11    parallelFastMarchingMethod()
12  Barrier
    // Move one level up
13  ChildLevel ← ParentLevel
14  ParentLevel ← ParentLevel.parentLevel()
15

```

Parent grid points covered by child grid points are corrected by evaluating the level-set function ϕ at the position of the parent grid point on the child level. ϕ is evaluated by linear interpolation.

Figure 2 provides a schematic representation of the different interpolation cases: There, the extracted position of the interface on the parent and child level is shown by the green lines, thick for the parent mesh and thin for the child mesh. Red dots are used for grid points located on one side of the interface and

² <https://www.openmp.org/>.

blue dots for the other side of the interface. All parent grid points are labeled with a letter from A to I. Not all child grid points are marked by a dot, because some are not computed by the FMM on the child level, due to narrow-band re-distancing.

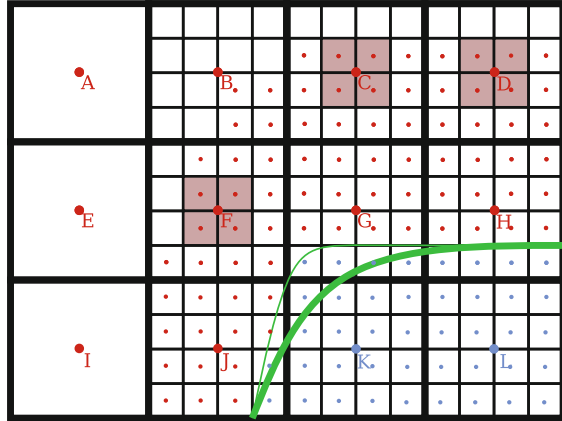


Fig. 2. Schematic representation of different cases for the interpolation of the parent level grid points.

The parent grid points G, H, J, K, and L have a neighbor with an opposite sign. They are marked `KnownFix` by the FMM and, therefore, must not be modified. The parent grid points A, E, and I are not covered by a fine mesh and thus cannot be interpolated. The parent grid point B is also not interpolated, because the narrow-band re-distancing on the child level did not compute all the necessary points for the interpolation. Finally, parent grid points C, D, and F are interpolated. For each of those grid points the linear interpolation is based on the four closest child grid points (marked by a red background).

The interpolated grid points are stored in separate *lists* for each child mesh. Subsequently, the lists are used for initializing the FMM. The mesh based parallelization approach presented in Sect. 3 is employed. An OpenMP task is created for all child meshes, since a parent grid point is refined by at most one child mesh (cf. Algorithm 1, Line 6).

The previously created lists for every child mesh with a common parent mesh are merged together. The lists are then used to initialize the FMM on the parent grid level. The actual initialization step, however, is modified compared to the original FMM. The grid points keep their distance and their flag from the previous re-distancing step. If a parent grid point is marked `KnownFix`, the correction is not applied, in order to avoid modifying the zero level-set (cf. Algorithm 1, Line 8). This is performed in parallel for all parent meshes with OpenMP tasks.

Then, the unmodified marching of the FMM is performed, correcting the solution in regions not covered by child meshes (cf. Algorithm 1, Line 10). The

parallelization of the marching step is the same as for the regular re-distancing discussed in Sect. 3. After the FMM has finished processing, it continues by moving up in the hierarchy until Level 1 has been corrected.

The proposed initialization of the FMM avoids the re-computation of all grid points. The original FMM initializations (cf. Sect. 2) would trigger a re-computation of all grid points and therefore double the runtime on each level.

The disadvantage of the proposed initialization is, that during the marching step (cf. Algorithm 1, Line 11) grid points, for which the distance to the interface was under-estimated are not corrected. This is inherent to the FMM, as grid points are only processed if the newly computed distance is smaller than the previously assigned one. This is a reasonable choice as the FMM tends to over-estimate the distance [16].

5 Computational Results and Analyses

5.1 Test Cases

The correction step is first evaluated on two-dimensional examples: *Corner*, *Sharp Corner*, and *Narrow Trench* (Fig. 3a, Fig. 3b, and Fig. 4, respectively). The error norms for the different levels are compared in Table 1, Table 2, and Table 3. Additionally, a *Three-Dimensional Trench* example inspired by a fabrication simulation in the field of microelectronics is given (cf. Fig. 5). The error norms, as the simulations compare to the exact solution, are shown in Table 4. Due to the computationally significant load induced by the three spatial dimensions, the parallel speed up is investigated.

The domain for all test cases is $[-1, 1]^d$, with $d \in \{2, 3\}$ and symmetric boundary conditions. The two-dimensional test cases are discretized by 40 grid points in each direction on the coarsest level. The three-dimensional test case is discretized using 50 grid points in each direction. In all test cases, two levels of refinement with a constant factor 4 in all spatial directions are used. In total, there are three levels. The initialization and the exact solution are geometrically computed from the triangle ($d = 3$) and line ($d = 2$) representations of the investigated geometries based on the Computational Geometry Algorithms Library (CGAL)³.

Benchmarking Platform

The benchmarking results have been obtained by using a single node of the Vienna Scientific Cluster 3 (VSC3)⁴. The node is equipped with two 8-core Intel Xeon E5-2650v2 processors, running at 2.6 GHz and 64 GB of DDR3 memory. The benchmark implementation is written in C++11 and compiled in a GNU/Linux environment using GCC-7.3 with optimization flag `-O3`.

³ <https://www.cgal.org/>.

⁴ <http://vsc.ac.at/>.

5.1.1 Corner Examples

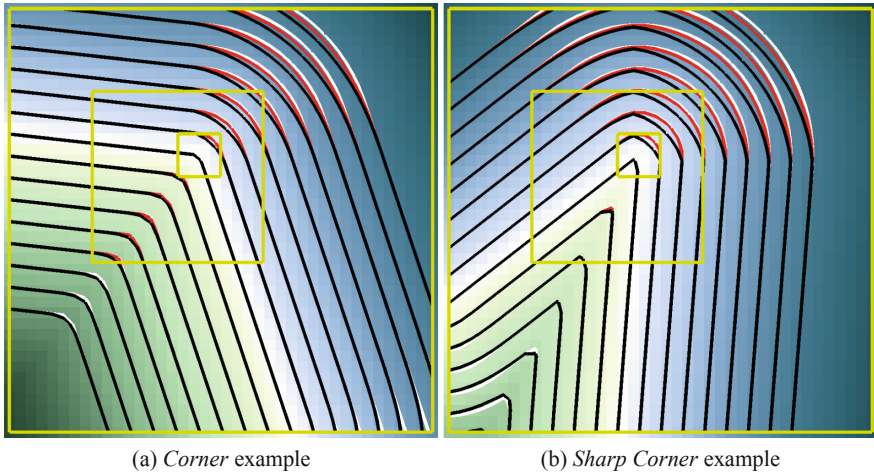


Fig. 3. Isolines on Level 1 (coarsest grid): Black lines denote the solutions based on re-distancing without the correction step, red lines with the correction step, and white lines for the exact solution. The green and blue background colors give the distance to the interface. The yellow boxes show the outline of the meshes, there is only one mesh on each level.

The corner test cases have the refined levels around the corner near the center of the domain, as shown by the yellow boxes. The symmetric boundary condition causes additional corners at the domain boundary. For these corners no refinement is employed, so that a correction is not possible (cf. Fig. 3a, lower right and Fig. 3b, lower left). Figure 3 shows the isolines extracted from Level 1. Notably, the corner is not represented by a single point as the corner is purposely not grid aligned, as is the usual case in practical level-set simulations. The FMM over-estimates the distance for rarefaction waves (reflex angle side) and under-estimates the distance at shock waves. This can be seen on the obtuse angle side in Fig. 3a and on the, acute angle side in Fig. 3b.

The proposed correction step reduces the error (cf. Table 1 and Table 2) in regions covered by a finer mesh and for the rarefaction wave. In the *Corner* example the errors in L1-norm and L2-norm are reduced by a factor of 2.1 on Level 1 and by a factor of 1.9 on Level 2. For the *Sharp Corner* example the reduction is even higher, 2.7 on Level 1 and 2.1 on Level 2, as sharper corners benefit more from the correction. On Level 3 the correction is not possible, as it is the finest level.

Table 1. Error norms for the *Corner* example, with and without the correction step applied, and the corresponding factor by which the error norm is reduced.

Level	L1-norm	L1-reduc	L2-norm	L2-reduc	inf-norm	inf-reduc
1	5.437e-3		3.260e-4		4.785e-2	
1 corrected	2.491e-3	2.2	1.550e-4	2.1	3.079e-2	1.6
2	1.122e-3		5.101e-5		1.393e-2	
2 corrected	6.035e-4	1.9	2.792e-5	1.8	8.541e-3	1.6
3	5.126e-4		1.819e-5		3.757e-3	

Table 2. Error norms for the *Sharp Corner* example, with and without the correction step applied, and the corresponding factor by which the error norm is reduced.

Level	L1-norm	L1-reduc	L2-norm	L2-reduc	inf-norm	inf-reduc
1	9.110e-3		4.823e-4		6.212e-2	
1 corrected	3.388e-3	2.7	1.812e-4	2.7	2.707e-2	2.3
2	1.894e-3		7.264e-5		1.753e-2	
2 corrected	8.957e-4	2.1	3.484e-5	2.1	9.546e-3	1.8
3	8.866e-4		2.569e-5		4.661e-3	

5.1.2 Narrow Trench

This test case consists of a narrow trench on a flat surface. The trench has a width of 0.01 (i.e., 0.5% of the domain dimension) whereas the grid spacing on Level 1 is only 0.05, thus too coarse to resolve the trench, as no grid points with opposite signs exist along the trench. The re-distancing algorithm without the correction step does not result in a trench, yielding only a small dent in the surface (cf. Fig. 4a). The resolution on the finer levels allows to resolve the trench (cf. Fig. 4b), thus the correction step enables its representation also on Level 1. The errors in L1-norm and L2-norm are reduced by a factor of 15.3 and 14.4, respectively for Level 1 but only by a factor of 1.6 for Level 2 (cf. Table 3). The substantial difference in the improvement between Level 1 and Level 2 is due to the fact that the trench is already representable on Level 2. This can be seen by comparing the black isolines in Fig. 4 for Level 1 and Level 2.

5.1.3 Three-Dimensional Trench

The three-dimensional example combines corners and a trench. There is a step and a trench with a width smaller than the grid resolution of Level 1 (cf. Fig. 5). The trench has a slight dent, so that it is not aligned with the grid. Geometrical configurations like this often appear in microelectronic devices [17], in particular, high aspect ratios are common [18].

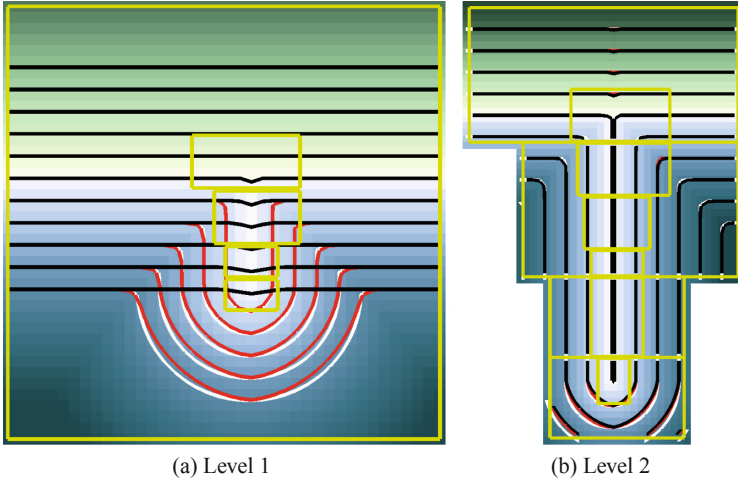


Fig. 4. Isolines for Level 1 and Level 2 for the *Narrow Trench* example. Black lines show the results based on re-distancing without the correction step, red lines with the correction step, and white lines for the exact solution. The green and blue background colors give the distance to the interface. The yellow boxes show the mesh configuration on Level 1 and Level 2 in (a) and Level 2 and Level 3 in (b).

Table 3. Error norms for the *Narrow Trench* example, with and without the correction step applied, and the corresponding factor by which the error norm is reduced.

Level	L1-norm	L1-reduc	L2-norm	L2-reduc	inf-norm	inf-reduc
1	9.101e-2		4.539e-3		4.839e-1	
1 corrected	5.941e-3	15.3	3.148e-4	14.4	4.005e-2	12.1
2	6.732e-4		5.222e-5		1.262e-2	
2 corrected	4.129e-4	1.6	3.339e-5	1.6	8.398e-3	1.5
3	6.709e-5		4.236e-6		3.400e-3	

The given mesh placement is visualized in Fig. 6 for the different levels. There are 107 meshes on Level 2 containing 674 496 grid points in total and on Level 3 there are 591 meshes with a total of 6 146 688 grid points.

In this test case the errors in the L1-norm and L2-norm for the corrected levels are smaller by a factor of at least 4.2 on Level 1 and 1.5 on Level 2. The reasons are similar to the ones presented in the *Narrow Trench* example. The inf-norm for the error on Level 2 has not decreased, since the grid point causing it is not covered by a refined mesh and is also located on a shock wave. The FMM cannot correct such errors, though a skeleton⁵ aware modification

⁵ A skeleton is the union of all points with more than one interface point closest to them [19].

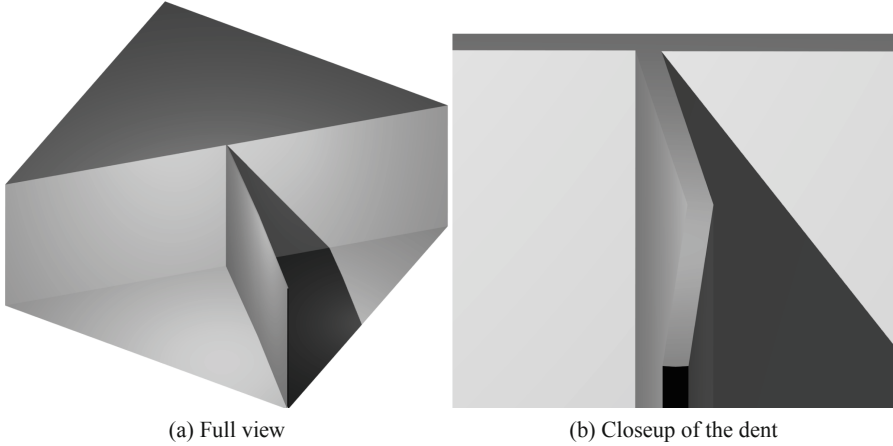


Fig. 5. Rendering of the *Three-Dimensional Trench* example (view from the bottom). The dent is only visible from a particular angle.

Table 4. Error norms for the *Three-Dimensional Trench* example, with and without the correction step applied, and the corresponding factor by which the error norm is reduced.

Level	L1-norm	L1-reduc	L2-norm	L2-reduc	inf-norm	inf-reduc
1	1.853e-2		1.356e-4		4.593e-1	
1 corrected	4.422e-3	4.2	2.644e-5	5.1	3.478e-2	13.2
2	2.426e-4		1.259e-6		1.418e-2	
2 corrected	1.588e-4	1.5	8.429e-7	1.5	1.418e-2	1.0
3	3.165e-5		7.340e-8		3.017e-3	

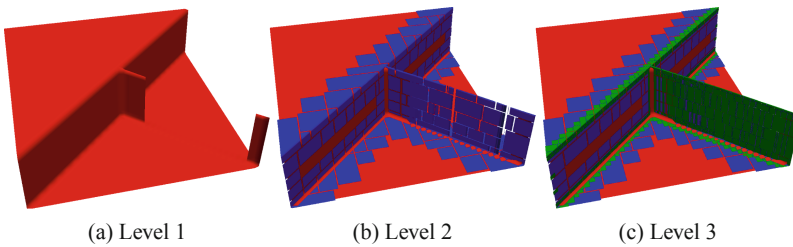


Fig. 6. Interface representation on the different mesh levels (view from the bottom). On Level 1 (red) the trench is too thin to be resolved by the spatial discretization. Level 2 (blue) and Level 3 (green) show the placement of the different meshes (small gaps in between).

of the FMM could potentially improve those grid points, by considering only grid points, which are not separated by the skeleton, for the calculation of the distance.

In Fig. 7a the runtime for re-distancing and the correction steps, together with the total runtime are shown. The correction step extends the re-distancing runtime by 4% for one thread and up to 10% for 16 threads. The increased runtime contribution for higher thread counts is caused by the limited parallel speed-up (cf. Fig. 7b) of the correction step.

A speed-up for the re-distancing step with correction of 9.3 has been achieved for 16 threads (cf. Fig. 7b), which corresponds to a parallel efficiency of 58%. The parallel speed-up for the correction step alone is inferior, because the non-parallelized Level 1 contributes more than half of the runtime for higher thread counts (cf. Fig. 8a). For a fair comparison (cf. Fig. 8b) the speed-up of the correction step is compared to the re-distancing on a per-level-basis for all levels. The speed-up strongly depends on the level, because the number and computational load of the meshes varies considerably among the levels.

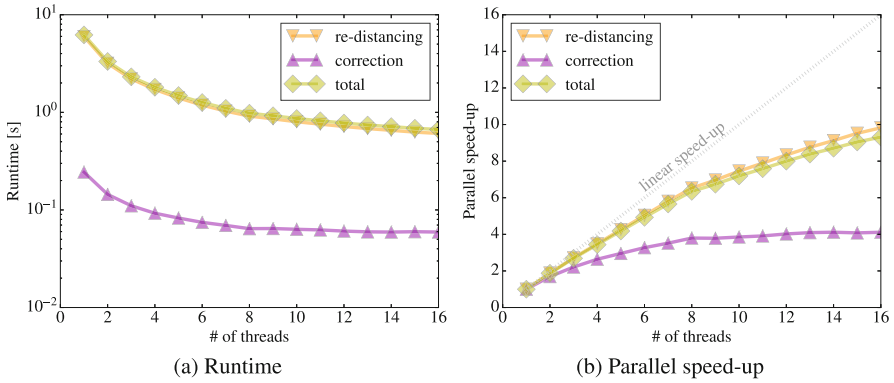


Fig. 7. Runtime and speed-up (all levels combined) for re-distancing and the correction step for up to 16 threads.

On Level 2 the correction step shows a speed-up of 7.5, which is nearly the same as for the re-distancing step (speed-up of 7.8). The speed-up is reduced because there is less computational load and, therefore, synchronization tasks take up proportionally more time. The second reason is that the initialization does not show any speed-up for more than 8 threads, due to non-uniform memory access (NUMA) effects.

On Level 1 there is practically no speed-up, because the mesh based parallelism cannot be applied for a single mesh. The shortest runtime for the initialization on Level 1 is achieved with 8 threads. With more threads the initialization step is limited by NUMA effects.

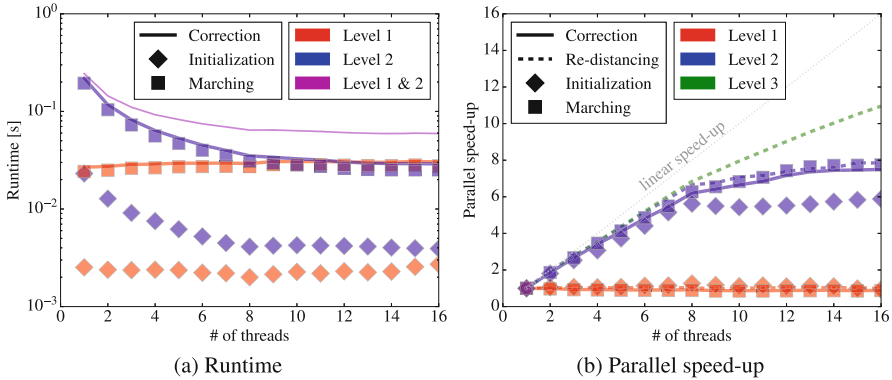


Fig. 8. Runtime and speed-up per level up to 16 threads. The speed-up on the same level is similar between the re-distancing and correction steps. The initialization and marching times are given for the correction step. The marching time dominates.

6 Conclusion

A bottom-up correction step for hierarchical re-distancing using the fast marching method has been presented. The correction step allows to represent trenches with a width smaller than the grid resolution on coarse grids and higher accuracy on rarefaction waves. Several test examples in two and three dimensions have shown a significant reduction of the error. The performance penalty of the correction step ranges from 4 to 10% of the original runtime. The total parallel speed-up of re-distancing with the correction step is 9.3 for 16 threads. Finally, we show that the level-by-level speed-up is similar to re-distancing without correction.

Acknowledgments. The financial support by the *Austrian Federal Ministry for Digital and Economic Affairs* and the *National Foundation for Research, Technology and Development* is gratefully acknowledged. The computational results presented have been achieved using the Vienna Scientific Cluster (VSC).

References

1. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci.* **93**, 1591–1595 (1996)
2. Gibou, F., Fedkiw, R., Osher, S.: A review of level-set methods and some recent applications. *J. Comput. Phys.* **353**, 82–109 (2018)
3. Cheng, L.-T., Tsai, Y.-H.: Redistancing by flow of time dependent Eikonal equation. *J. Comput. Phys.* **227**, 4002–4017 (2008)
4. Adalsteinsson, D., Sethian, J.A.: A fast level set method for propagating interfaces. *J. Comput. Phys.* **118**, 269–277 (1995)
5. Russo, G., Smereka, P.: A remark on computing distance functions. *J. Comput. Phys.* **163**, 51–67 (2000)

6. Detrixhe, M., Gibou, F., Min, C.: A parallel fast sweeping method for the Eikonal equation. *J. Comput. Phys.* **237**, 46–55 (2013)
7. Jeong, W.-K., Whitaker, R.T.: A fast iterative method for Eikonal equations. *SIAM J. Sci. Comput.* **30**, 2512–2534 (2008)
8. Weinbub, J., Hössinger, A.: Comparison of the parallel fast marching method, the fast iterative method, and the parallel semi-ordered fast iterative method. *Procedia Comput. Sci.* **80**, 2271–2275 (2016)
9. Royston, M., Pradhana, A., Lee, B., Chow, Y.T., Yin, W., Teran, J., Osher, S.: Parallel redistancing using the Hopf-Lax formula. *J. Comput. Phys.* **365**, 7–17 (2018)
10. Yang, J., Stern, F.: A highly scalable massively parallel fast marching method for the Eikonal equation. *J. Comput. Phys.* **332**, 333–362 (2017)
11. Weinbub, J., Hössinger, A.: Shared-memory parallelization of the fast marching method using an overlapping domain-decomposition approach. In: *Proceedings of the 24th High Performance Computing Symposium*, pp. 1–8 (2016)
12. Diamantopoulos, G., Weinbub, J., Selberherr, S., Hössinger, A.: Evaluation of the shared-memory parallel fast marching method for re-distancing problems. In: *Proceedings of the 17th International Conference on Computational Science and Its Applications*, pp. 1–8 (2017)
13. Diamantopoulos, G., Hössinger, A., Selberherr, S., Weinbub, J.: A shared memory parallel multi-mesh fast marching method for re-distancing. In: *Advances in Computational Mathematics*, pp. 1–17 (2019). <https://doi.org/10.1007/s10444-019-09683-z>
14. Joppich, W., Mijalković, S.: *Multigrid Methods for Process Simulation*. Computational Microelectronics. Springer, Vienna (1993)
15. Dijkstra, E.W.: A note on two problems in connexion with graphs. *Numer. Math.* **1**, 269–271 (1959)
16. Popovici, A.M., Sethian, J.A.: 3-D imaging using higher order fast marching traveltimes. *Geophysics* **67**, 604–609 (2002)
17. Radjenović, B., Lee, J.K., Radmilović-Radjenović, M.: Sparse field level set method for non-convex hamiltonians in 3D plasma etching profile simulations. *Comput. Phys. Commun.* **174**, 127–132 (2006)
18. Liu, P., Zhang, D., Guo, J., Wang, W., Yang, F.: Optimization of photoresist development and DRIE processes to fabricate high aspect ratio Si structure in 5 nm scale. *J. Micromech. Microeng.* **29**, 035006 (2019)
19. Cornea, N., Silver, D., Min, P.: Curve-skeleton applications. In: *Proceedings of IEEE Visualization*, pp. 95–102 (2005)