# Physical-Space Refraction-Corrected Transmission Ultrasound Computed Tomography Made Computationally Practical

[1]Shengying Li, [1]Klaus Mueller, [2]Marcel Jackowski, [3]Donald Dione, [4]Lawrence Staib

[1]Stony Brook University, Stony Brook, NY [2]University of Sao Paulo, Sao Paulo, Brazil
[3]Ultrasound Detection Systems LLC. Madison, CT [4]Yale University, New Haven, CT
contact author: mueller@cs.sunysb.edu

**Abstract.** Transmission Ultrasound Computed Tomography (CT) is strongly affected by the acoustic refraction properties of the imaged tissue, and proper modeling and correction of these effects is crucial to achieving high-quality image reconstructions. A method that can account for these refractive effects solves the governing Eikonal equation within an iterative reconstruction framework, using a wave-front tracking approach. Excellent results can be obtained, but at considerable computational expense. Here, we report on the acceleration of three Eikonal solvers (Fast Marching Method (FMM), Fast Sweeping Method (FSM), Fast Iterative Method (FIM)) on three computational platforms (commodity graphics hardware (GPUs), multi-core and cluster CPUs), within this refractive Transmission Ultrasound CT framework. Our efforts provide insight into the capabilities of the various architectures for acoustic wave-front tracking, and they also yield a framework that meets the interactive demands of clinical practice, without a loss in reconstruction quality.

**Keywords:** Transmission Ultrasound Mammography, GPU Acceleration

## 1. Introduction

Compared to traditional X-ray breast mammography, ultrasound has the important advantage that no radiation dose is delivered to the patient. In addition, tomographic methods, as opposed to single projections of a compressed breast, have the advantage that lesions can be better localized in space. A modality combining all of these advantages is transmission ultrasound CT (UCT). Here, the patient submerges the breast into a water bath (which is far more convenient than the compression needed for X-ray breast mammography) and a ring transducer array (with 100s of elements) axially positioned around the breast acquires the time-of-flight and attenuation data used for CT reconstruction, one set of receiver data for each emitter position. For 3D reconstruction, one may either use a multi-ring device, or one may axially translate or spiral the ring and repeat acquisition. Such systems have been described [6].

The fact that acoustic rays suffer from severe refraction effects in the imaged tissue has hampered the direct use of traditional analytical back-projection or iterative projection/back-projection frameworks to achieve a practical UCT solution. Existing approaches so far have been unable to model the resulting curved rays efficiently and accurately, and this has limited the image resolution that can be achieved. While recent work in iterative UCT [5] shows promise and discusses reflection and refraction effects, the reconstruction algorithms discussed there are still limited to the straight-ray assumption. In earlier work, several researchers (e.g. [2]) have explored the use of general ray-tracing, via ray-linking and ray re-binning, in P/BP UCT to improve the image quality. Denis [4] compared several methods for ray-tracing, showing that substantial improvement over straight ray methods can be achieved for moderately refracting fields. Yet, the more recent work typically avoids the expensive direct ray simulation. For example, Quan [11] reports that a simulation of 256 detectors would take weeks to complete on a desktop PC, requiring supercomputers for UCT reconstruction. They therefore use a sparse matrix solver to accomplish the reconstruction in reasonable time. Finally, an alternative method for UCT is diffraction tomography [9]. It is based on the weak scattering assumption, which, however, is violated in case of the breast, due to its strongly refracting fat layers.

Essentially, sparse matrix solver-based and Fourier-space methods do not model the ray physics in a direct manner, but through some intermediate representation, which cannot easily capture the local interactions of the ray with the medium. On the other hand, a complete simulation of the wave equation is prohibitively expensive. Yet, we [8] have shown that fast wave-front propagation methods, such as the Fast Marching Method (FMM) [12], can model the governing Eikonal equation well, taking into account refraction effects and allowing the resulting curved rays to be accurately modeled for both speed and attenuation in their native physical domain. We aim to resolve lesions of single pixel-size (~ data resolution), various shapes (speculated, lobulated, sharp, circumscribed), and a pixel-level distance among lesion clusters. At a US frequency of 7.5MHz one can reach depths of 16cm, exceeding the requirements for breast imaging, allowing a sub-mm resolution of less than 0.3mm.

While the FMM-based approach is far more efficient than a full wave equation simulation, we find that it is still not efficient enough to warrant deployment in a clinical setting. Thus, we look into the use of commodity high-performance hardware for parallel program execution, such as multi-core CPUs, programmable commodity graphics hardware (GPU), and processor clusters. However, the inherently sequential mechanism of the FMM method poses certain trade-offs with regards to the architecture used (and also cost), which we will show can be balanced by use of other equivalent fast wave front tracking algorithms, such as the Fast Sweeping Method (FSM) [3] and the Fast Iterative Method (FIM) [7]. These considerations and their results are the topic of this paper, with the overall conclusion being that UCT based on fast wave-front tracking has excellent potential for real life clinical deployment

## 2. Methods

The algorithm put forward in [8] uses SART (Simultaneous Algebraic Reconstruction Technique) [1]as the basis of their UCT reconstruction algorithm:

$$c_i = (p_i - \sum_j v_j^k w_{ij}) / \sum_j w_{ij} \qquad v_j^{k+1} = v_j^k + \lambda \sum_i c_i w_{ij} / \sum_i w_{ij} \tag{1}$$

Here, the $p_i$ are the measured receiver data, the $v_j$ are the reconstructed lattice voxels, the $w_{ij}$ are the (interpolation) weights relating a traversing ray originating at pixel $p_i$ to a voxel $v$, and $\lambda$ is a relaxation factor. The $c_i$ are the correction factors computed from the forward projection and delivered to the voxels in the back-projection step.

The propagation of the US wave front in 3D is modeled by the Eikonal equation:

$$(\partial t / \partial x)^2 + (\partial t / \partial y)^2 + (\partial t / \partial z)^2 = 1 / F^2(x,y,z) \tag{2}$$

Here, the speed term $F$ is a measure of the local sound conductance properties. To solve the equation for $F$ one must find the speed trajectory for each ray that minimizes the time cost function from the source. This is iteratively converged to using the SART. The forward projection computes for every voxel $(x,y,z)$ the time $T(x,y,z)$ at which the sound wave, originating from the emitter, has traversed it. We call this the time-of-flight (TOF) image, which includes the TOF estimates of the receivers themselves (used to correct the grid corrections). This TOF image allows a local computation of the ray direction vectors from all receivers back to the single emitter using the TOF image gradients, estimated via a high-quality filter. This crucial step ensures that a given curved ray will not miss the emitter, eliminating the path assembly overhead of earlier approaches. More formally, this *Eikonal-SART* algorithm is described as follows (see also Fig. 1 which assumes an emitter/detector ring array):

1. Construct the TOF field as described above.
2. Compute the corrections $c_i$ by subtracting the Eikonal-computed TOF field at the receivers from the true TOF data. Calculate the normalization weight by tracing curved rays back to the emitter using the TOF image and a field of unit voxels.
3. Update the speed terms of the voxels $v_{ij}$ by the reciprocal of the $c_i$, again using the TOF image to determine the curved ray paths.
4. Normalize the $v_{ij}$ by the weights (also accumulated in step 3).
5. Randomly select another (unused in this iteration) emitter and return to step 1.

In [8] we also described a data-driven relaxation factor to control the voxel updates, which yields superior results over the fixed relaxation factor $\lambda$ typically used in SART. In fact, our UCT reconstructs two tissue properties, which gives non-redundant information. Once the sound velocity (SV) image is reconstructed it is utilized to guide the non-linear rays for reconstructing the sound attenuation image using attenuation data.

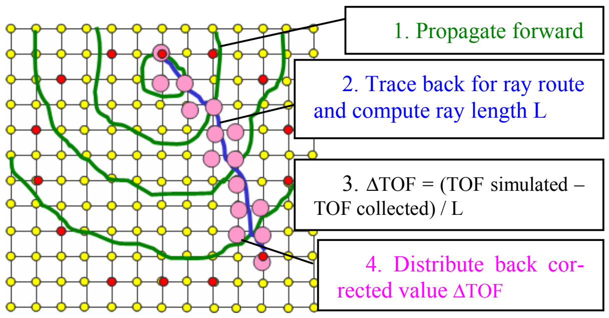
1. Propagate forward
2. Trace back for ray route and compute ray length L
3. ΔTOF = (TOF simulated – TOF collected) / L
4. Distribute back corrected value ΔTOF

**Fig. 1.** The Eikonal-SART refractive UCT algorithm.

## 2.1 How to perform refractive UCT efficiently: general considerations

Each iteration of Eikonal-SART performs for each of the $M$ emitters: (i) one forward wave propagation across a lattice of $N^3$ voxels, and (ii) one backward projection delivering the updates to the $N^3$ voxels via $M$-1 non-linear rays which are guided by the

TOF field computed in step (i). Let us assume we use $M=N^2$, that is, $N$ transducers for each of the $N$ rings (we shall neglect for now that the reduced axial dimension of the breast would allow far less rings to be used in practice). This makes for a slightly underdetermined equation system which has been shown to be sufficient in iterative reconstruction scenarios. For the raytracing in the back-projection step, we have $N^2$ rays of length O($N$) each, but the traversal involves only local and independent computations at each step, at low computational overhead. Thus the overall complexity for one back-projection is O($N^3$). The wave-front tracking in the forward projection also updates $N^3$ voxels to yield the TOF image (and advances the front), but the selection of the advance-voxel has complexity O($N^3$) since one must consider all voxels within the wave-front's narrow band (which may be arbitrarily complex). Thus the complexity for wave front tracking is O($N^6$).

Raytracing is quite amenable to parallel implementation and speedups on the order of 1-2 magnitudes can be obtained on various parallel platforms, such as GPUs (e.g. [10]). The independence of the individual rays affords a relatively fine-grained parallelism. In contrast, the wave front tracking does not exhibit much parallelism, since it is an inherently sequential algorithm where the selection of a voxel into the front depends on all previous selections. One method to reduce the wave front tracking overhead per emitter is to divide the 3D grid into $N$ 2D slices. This strategy gives good parallelism on a coarse-grained level, due to the independence among slices. There is no communication overhead between slices, no need for data distribution as each slice can reside on a separate processor, and no synchronization delay because each computation proceeds independently. While the 2D slice-based reconstruction may lead to errors due to out-of-plane wave propagation, our experiments indicate (see section 3) that the reconstructions using the 2D slice decomposition are quite similar to the corresponding fully-3D reconstruction, but can be obtained at much higher speeds. But even with this decomposition there are vast differences in the level of speedup obtained with different Eikonal equation solvers (all using physical-space wave front tracking), also as a function of computing platform, as is discussed next.

## 2.2 Numerical Eikonal equation solvers and their parallelization potential

**The Fast Marching Algorithm (FMM) algorithm** [12] solves the Eikonal equations in one pass, tracking the evolution of an expanding front. A key observation of FMM is that the solution of Eq. (2) at each grid point depends only on the smaller values of neighboring points. Thus the solution of the equation can be built in the order of increasing arrival time of points. Here it is essential to quickly locate the voxel with the smallest value in the active "narrow-band" of $n$ voxels. Typically a heap is used for this, with an update (insertion) time of O($logn$). Sapiro [14] proposed a faster algorithm using an *untidy priority queue*, which replaces the heap's tree by a table, reducing the update complexity to O(1). This, however, also introduces extra errors caused by misorderings inside the queue. We employed the (min-)heap data structure for accuracy, using double-linking between lattice and heap for speed.

The FMM is not naturally accelerated on fine-grained parallel systems. Neither the min-heap data structure nor the priority queue, which enables fast speeds on the CPU, encourages parallelism, since they all sequentialize computations and make them globally dependent. The slice decomposition described in section 2.1 favors coarse-grained parallelism, assigning projections of different slices to different threads or processors and running them concurrently. GPUs, programmed with CUDA, are well suited to compute this type of problem, but one must be aware of the large cache memory requirements of the fast min-heap operations. Section 3 shows that this poses a limiting factor for GPUs, where the register, cache and global memory are limited.

**The Fast Sweeping Method (FSM) algorithm** [15] applies Gauss-Seidel iterations with alternating sweep orderings. Key is to follow the solution characteristics and update points in that order, *i.e.*, the sweeping direction should fit the real information propagation. In two dimensions, the sweep order is according to these four loops:

$$i{:}1 \,..\, nx,\; j{:}1 \,..\, ny \;\|\; i{:}\, nx \,..\, 1, j{:}\, 1 \,..\, ny \;\|\; i{:}\, nx \,..\, 1,\; j{:}ny \,..\, 1 \;\|\; i{:}\, 1 \,..\, nx,\; j{:}\, ny \,..\, 1$$

where $i$ and $j$ are the sweeping point positions in the $x$- and y-directions, respectively, and $nx$ and $ny$ are the number of points in the $x$-direction and $y$-direction, respectively.

The FSM is amenable to parallelization, since it follows the causality along the solution characteristics in a parallel manner. It can compute multiple sweeping directions simultaneously. In addition, it can divide the whole domain into sub-domains and execute sub-domain computations in parallel. It is well suited for multi-processor CPU systems or clusters. However, since in each sweep thread in a sub-domain the computation is sequential, it is not easy to achieve fine-grained parallelism.

**The Fast Iterative Method (FIM) algorithm** [7] solves PDEs of H-J (Hamilton-Jacobi) equations on parallel architectures. The Eikonal equation is a special case of the H-J equation. It uses Godunov upwind discretization of the Hamiltonian [13] and updates the value on a narrow band iteratively. The FIM allows multiple updates per point, by re-inserting points into the narrow band, called *active list AL*.

The FIM gives values of zero to the source and $\infty$ to all other points. All neighbors of the source points form the AL. In the update loop, the FIM calculates the Godunov Hamiltonian for all AL points and updates their distance values if the newly calculated value is less than the current. If an AL point has converged, it is removed from the AL. All neighbors of the AL points will also update and if their value is less than the current, they will be added to the AL. This continues until the AL is empty.

This algorithm maps well onto GPUs with parallel SIMD processors due to its fine-grained level parallelism, updating all AL points in parallel. In order to fully use the coherent memory access and control flow, our GPU implementation uses data-grouping as the primitive execution unit. Each point inside a data-group is executed as a thread, all threads for the same data-group combine into a block, and all blocks combine into the CUDA computational domain as a grid. The CPU-algorithm AL becomes the Active Block List (ABL) on the GPU. In each iteration all active blocks are updated and converged blocks are removed from the ABL. Neighbor blocks are also updated. Further, since points inside a data-group share the same CUDA block, they use shared memory simultaneously. Shared memory is on-chip memory at CPU register-speed and is much faster than global memory. In order to fully exploit fast shared memory access, we coalesced the memory by assigning points in the same data-group (a 2D block of 8×8 slice pixels) next to neighboring points in memory.

## 3 Results

All reported results use the Eikonal-SART reconstruction scheme presented above. All quality assessments are based on the global $L_2$ RMS error (ultrasound phantom vs. reconstructed densities). We created a physically-based human breast ultrasound phantom, based on the cryosection color images of the NIH Visible Female dataset. Hue was used to identify the basic tissues and their acoustical mapping was obtained by consulting biophysics tables (see Table 1). A few lesions were also added. The phantom size is $128^2 \times 40$ and the smallest lesion is a sphere of a two-voxel diameter.

**Table 1.** Breast phantom ultrasound properties.

| Property | Tissue | Fat | Lesions (LG) | Lesions(SM) | Skin |
|---|---|---|---|---|---|
| Velocity | 1475 m/s | 1375 m/s | 1560 m/s | 1530 m/s | 1650 m/s |
| Attenuation | 50 | 15 | 60/30 | 70 | 58 |

**Reconstruction quality:** We employed a high-quality wave equation solver to generate the projection data. Reconstructions (with 256 detectors per slice) were performed with the three numerical Eikonal equation solvers presented above (FMM, FSM, and FIM). All achieved nearly the same quality (3-4% RMSE). This is a vast improvement over a straight-ray reconstruction (25% RMSE) and demonstrates that accurate refraction modeling within a physically-based ray-tracing scheme enables reconstructions at high fidelity and accurate geometry. Fig. 2 gives visual evidence (for sound velocity SV) that the results are in excellent agreement with the original (acoustically mapped) phantom, with small detail being recovered well. Fig. 3 (a) shows the attenuation reconstruction result (7.5% RMSE). Fig 3 (b) demonstrates that the 2D slice-by-slice reconstruction (right, RMSE 3.6%) can reach nearly the same quality than a fully-3D reconstruction (center, RMSE 3.4%). Here we observe that the small-

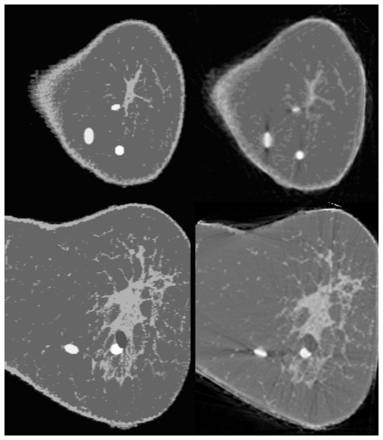
**Fig. 2.** (Left) acoustic breast phantom derived from the Visible Female data-set and (right) SV reconstructions. First row: slice close to the center, second row: slice close to the bottom.

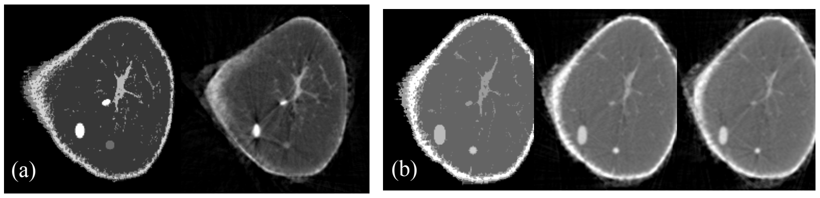

**Fig. 3.** (a) Attenuation image: (left) phantom, (right) reconstructed. (b) Fully-3D vs. 2D slice-based reconstruction (left) phantom, (center) fully-3D, (right) 2D slice-based.

est lesions (of two-voxel diameter close to the center) can be clearly identified in both reconstructions. Experiments with noisy data show that even with SNR=10 (considered challenging) the inner lesions/structures remain very visible and distinguishable.

**Time performance:** With FMM we were able to reconstruct a $128^2$ image from 256 transducers in about 60 seconds (3 iterations) on a 2.8GHz Pentium 4. For the 3D case, on a $128^3$ grid, it takes 9.76 sec for a single wave propagation from one emitter. A full reconstruction with 256×128 transducers would then take about 533 hours. In contrast, the 2D slice decomposition approach would finish in 125 min, since the time per 2D wave tracking decreases to 0.03 sec. But this is still too slow for practical use.

We then implemented FMM in CUDA, using an NVIDIA 8800 GTX GPU with 768 MB RAM and an NVIDIA Tesla with 1.5 GB RAM. To make full use of the GPU multiprocessors, it is essential to run a large number of concurrent threads. However, we were unable to beat CPU performance, even with a large number of concurrent threads. One reason is slow global memory access. Different threads do not share memory, and every thread requires frequent (and expensive) access to global memory. Another reason is that each thread kernel uses 28 registers (total is 8,192) for the narrow band variables. This impeded the allocation of a sufficient number of threads, yielding a mere 33% multiprocessor occupancy.

For the FSM, our CPU implementation took 0.031s (0.18s) for one projection (wave front tracking) of a $128^2$ ($256^2$) image. When run on a quad-core computer, a 3.5-fold speedup is achieved. On a distributed memory computer cluster with 8 nodes and 4 sub-domains and parallel sweep directions, we gained a 5-times speed up. Our FIM-implementation on CPU required 0.04s (0.19s) for one projection of a $128^2$ ($256^2$) image. The small-grain parallelism of the FIM favors an extension to multi-core, multi-processor, and GPU-SIMD configurations. The GPU implementation required only 0.002 sec for one projection of a $256^2$ image, which is about 80 times faster than the fastest algorithm on the CPU (the FMM). Thus, for an object of size $256^2 \times 44$ and 512 transducers in each ring, it requires less than 5 minutes to reconstruct an object, which is a time that can easily satisfy real-life clinical scenarios.

The first two rows of Table 2 compare the time required for a single projection (wave front tracking) for two grid sizes and the second two rows compare the time required for 3D reconstruction using these three methods. We see that on the CPU, the one-pass approach of the FMM has the best performance, aided by the min-heap data structure. The FSM works well for multi-core and multiprocessor clusters (8 for projections and for reconstruction) since it can be computed in parallel, with little data communication. But clusters are expensive – an 8-node state-of-the-art computer cluster might cost thousands of US dollars. The GPU provides a good balance of

**Table 2.** Time (in s) required for one projection (wave propagation) for various Eikonal solvers (first 2 rows) and for a 3D reconstruction using the Eikonal-SART algorithm (second 2 rows)

| Task | Grid size | FMM | FSM | | FIM | | |
|---|---|---|---|---|---|---|---|
| | | CPU | CPU | Cluster-8 nodes | CPU | GPU (NVIDIA) | |
| | | | | | | 8086GTX | Tesla |
| Projection | $128^2$ | 0.025 | 0.031 | 0.007 | 0.038 | 0.00082 | 0.00072 |
| Projection | $256^2$ | 0.097 | 0.180 | 0.039 | 0.189 | 0.0023 | 0.0022 |
| Reconstr. | $128^2 \times 40$ | 2400 | 2649 | 506 | 3316 | 37.8 | 30.9 |
| Reconstr. | $256^2 \times 44$ | 19200 | 32732 | 6393 | 37221 | 286.2 | 225.76 |

price and performance. The FIM performs best on the GPU due its fine-grained parallelism. A NVIDIA Tesla costs only $600, yet it achieves about an 80-fold speedup over the fastest CPU FMM implementation.

## 4 Conclusions

We have demonstrated that iterative 3D UCT reconstruction with proper refraction physics, modeled directly in the object domain via wave front tracking can be accomplished at clinical rates. The reconstructions obtained from realistic phantoms resolve fine details well. Future work will further our quality assessments, also with real data

**Acknowledgments.** This research was supported by NIH grant R21 EB004099-01.

## References

1. Andersen, A., Kak, A.: Simultaneous Algebraic Reconstruction Technique (SART): A Superior Implementation of the ART Algorithm. Ultrasound Imaging, 6, 81--94 (1984)
2. Andersen, A.: A Ray Tracing Approach to Restoration and Resolution Enhancement in Experimental Ultrasound Tomography. Ultrasound Imaging, 12, 268--291 (1990)
3. Boue, M., Dupuis, P.: Markov Chain Approximations for Deterministic Control Problems with Affine Dynamics and Quadratic Costs in the Control. SIAM J. Numerical Analysis, 36, 667--695 (1999)
4. Denis, F., Basset, O., Gimenez, G.: Ultrasonic Transmission Tomography in Refracting Media: Reduction of Refraction Artifacts by Curved-ray Techniques. IEEE Trans. Medical Imaging, 14, 173--88 (1995)
5. Duric, N., Littrup, P., Babkin, A. et al.: Development of Ultrasound Tomography for Breast Imaging: Technical Assessment. Medical Physics, 32(5), 1375--1386 (2005)
6. N. Duric, P. Littrup, et al.: Detection of Breast Cancer with Ultrasound Tomography: First Results with the Computerized Ultrasound Risk Evaluation (C.U.R.E) Prototype. Medical Physics, 34(2), 733--785 (2007)
7. Jeong, W., Fletcher, P., Tao, R., Whitaker, R.: Interactive Visualization of Volumetric White Matter Connectivity in DT-MRI Using a Parallel-hardware Hamilton-Jacobi Solver. IEEE Trans. Visualization and Computer Graphics, 13(6), 1480--1487 (2007)
8. Li, S., Mueller, K., Jackowski, M., et al.: Fast Marching Method to Correct for Refraction in Ultrasound Computed Tomography. In: IEEE Symp.Biomed. Img. 896--899, (2006)
9. Pan, X.: Unified Reconstruction Theory for Diffraction Tomography, with Consideration of Noise Control. J. Optical Society America A. 15, 2312--2326 (1998)
10. Purcell, T., Buck, I., Mark, W., Hanrahan, P.: Ray Tracing on Programmable Graphics Hardware. ACM Trans. Graphics, 21(3), 703--712 (2002)
11. Quan, Y., Huang, L.: Sound-speed Tomography Using First-arrival Transmission Ultrasound for a Ring Array. In: Proc. SPIE Medical Imaging, 6513 (2007)
12. Sethian, J.: Level Set Methods and Fast Marching Methods. Cambridge Monographs on Applied and Computational Mathematics, 2nd Ed. (1999)
13. Sethian, J., Vladimirsky, A.: Ordered Upwind Methods for Static Hamilton-Jacobi Equations: Theory and Algorithms. SIAM J. of Numerical Analysis, 41(1), 325--363 (2003)
14. Yatziv, L., Bartesaghi, A., Sapiro, G.: O(N) Implementation of the Fast Marching Algorithm. Journal of Computational Physics, 212, 393--399 (2006)
15. Zhao, H.: Fast Sweeping Method for Eikonal Equations. Mathematics of Computation, 74, 603--627 (2005)