Proceedings of ALGORITMY 2016
pp. 204–213

HIGHLY EFFICIENT SURFACE NORMAL INTEGRATION

MICHAEL BREUß *, YVAIN QUÉAU †, MARTIN BÄHR *, AND JEAN-DENIS DUROU †

Abstract. The integration of surface normals for the computation of a surface in 3D space is a classic problem in computer vision. However, even nowadays it is still a challenging task to device a method that combines the flexibility to deal with non-trivial computational domains with high accuracy, robustness and computational efficiency. In this paper we propose to use for the first time in the literature Krylov subspace solvers as a main step in tackling the task. While these methods can be very efficient, they may only show their full potential when combined with a numerical preconditioning and even more importantly, a suitable initialization. To address the latter issue we propose to compute this initial state via a recently developed fast marching integrator. Numerical experiments prove the benefits of this novel combination.

Key words. surface normal integration, Poisson integration, Krylov subspace methods, fast marching methods

AMS subject classifications. 65F10, 65N06, 68U10

1. Introduction. The integration of surface normals is a fundamental task in computer vision. Classic examples for processes where this technique is often applied are shape from shading [11] and photometric stereo (PS) [24]. In these applications the depth of an unknown surface in 3D space is computed in a first step in terms of a field of surface normals, or equivalently, a corresponding gradient field. In a subsequent step this needs to be integrated in order to obtain the shape of the surface.

Let us briefly elaborate on the demands on an ideal integrator. As discussed e.g. in [6] a practical issue is the robustness with respect to noise and outliers. Since computer vision processes such as PS rely on simplified assumptions that often do not capture realistic illumination and light reflectance, such artefacts may often arise when estimating surface normals at hand of real-world input images. Secondly, objects of interest for 3D reconstruction are typically in the centre of a photographed scene and take only a portion of an input image. The sharp gradient representing the transition from foreground to background is a difficult feature for most surface normal integrators, as are discontinuous surfaces that may appear at self-occlusions of an object [2]. Therefore it is desirable to consider only image segments that represent the object of interest and not the background. With respect to reconstruction quality and efficiency, an ideal solver for surface normal integration should thus work on non-rectangular domains. Finally, as camera technology evolves, the resolution of images tends to increase continually. Consequently the computational efficiency of a solver is a key requirement for future applications.

In order to solve the problem of surface normal integration many methods have been developed. Important classic examples are the method of Horn and Brooks [11] which employs the calculus of variations, the direct line-integration scheme of Wu and Li [25] and the frequency domain method of Frankot and Chellappa [4]. Many improvements based upon these methods have been proposed, see e.g. [12, 13, 22] for a more detailed account. Among these techniques variational methods offer a

*Applied Mathematics Group, BTU Cottbus-Senftenberg, Platz der Deutschen Einheit 1, 03046 Cottbus, Germany

†IRIT - 2 rue Charles Camichel, Touluse Cedex 7, France

high robustness with respect to noise and outliers. Therefore this class of integrators has been the subject of many extensions in modern works, as e.g. by Durou and his co-workers [2, 3] accompanied by the use of alternative optimisation schemes [17] and the scheme of Harker and O'Leary [7]. However, while the latter scheme is computationally efficient it relies on a rectangular computational domain whereas the other variational methods are not too efficient in the non-rectangular case.

Our contributions. In this paper we propose a novel numerical scheme which explores as done in many works on the variational approach the associated Poisson equation. In detail, our contributions are: *(i)* We suggest to our knowledge for the first time in the computer vision literature to apply preconditioned Krylov subspace methods [15, 19] for solving the Poisson integration problem over non-trivial computational domains on a grid. *(ii)* For computing a good initialisation for the Krylov solver we propose to employ a recently developed fast marching (FM) integrator [5]. The main advantages of the FM integrator is its flexibility for use with non-trivial domains coupled with a low computational complexity. *(iii)* By a thorough experimental evaluation we show that our resulting novel method combines the known advantages of *flexibility* and *robustness* of variational methods with *low computational times* and *low memory requirements*.

2. The FM-Krylov Integrator. We now present the building blocks of our novel algorithm in two steps, first the FM integrator and then the Poisson solver.

2.1. The Fast Marching Integrator. For the purpose of using the FM scheme, a useful set-up for the normal integration problem can be described as follows. For a domain Ω a normal field $\mathbf{n} := \mathbf{n}(x,y) = (n_1(x,y), n_2(x,y), n_3(x,y))^\top$ is given at each grid point $(x,y) \in \Omega$. The task is to recover a surface S, which can be represented as a depth map $v(x,y)$ over $(x,y) \in \Omega$, such that $\mathbf{n}$ is a normal field of v. A normal field $\mathbf{n}$ of a surface at $(x,y,v(x,y))^\top$ can be written as

$$\mathbf{n}(x,y) := \frac{(-v_x, -v_y, 1)^\top}{\sqrt{\|\nabla v\|^2 + 1}} \qquad \text{with} \qquad v_x := \frac{\partial v}{\partial x},\ v_y := \frac{\partial v}{\partial y}\,. \tag{2.1}$$

This follows from considering the tangent vectors $(0,1,v_y)^\top$ and $(1,0,v_x)^\top$ to S, making use of the condition that a normal must be parallel to their vector product. Moreover the components of $\mathbf{n}$ are the partial derivatives of v with

$$\begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} -\frac{n_1}{n_3} \\ -\frac{n_2}{n_3} \end{pmatrix} =: \begin{pmatrix} p \\ q \end{pmatrix} \tag{2.2}$$

where we think of p and q as given data. Via $\mathbf{g}(x,y) := (p(x,y), q(x,y))^\top$, we obtain from (2.1) and (2.2) the condition $\nabla v(x,y) = \mathbf{g}(x,y)$. Taking the Euclidean norm on both sides this leads to the following eikonal-type partial differential equation (PDE):

$$\|\nabla v\| = \sqrt{p^2 + q^2} =: 1/F \tag{2.3}$$

where $F > 0$. In order to be able to solve (2.3) with FM a boundary condition $v(x_0, y_0) = v_0$ must be known.

The Model for FM Integration. It is important to note that we do not reconstruct the surface exactly from the latter equation. In order to prevent the situation $\sqrt{p^2 + q^2} = 0$ which would hinder the use of the FM method, we follow an idea of Ho *et al.* [10] and modify the latter PDE. To this end we aim to solve for a new function

w of the form $w := v + \lambda f$ with a parameter $\lambda > 0$ and a user-defined function f, so that the new unknown function w has a known minimum in (x_0, y_0).

In this setting, we do not compute v directly, but instead we solve in a first step for w. A natural candidate for f is the squared Euclidean distance function with the minimum in the centre of the domain $(x_0, y_0) = (0, 0)$, so we write

$$f := f(x, y) = x^2 + y^2. \tag{2.4}$$

Making the connection to the eikonal-type equation (2.3) we obtain the new constituting equation

$$\|\nabla w\| = \sqrt{(p + \lambda f_x)^2 + (q + \lambda f_y)^2} \tag{2.5}$$

where $\nabla v = (p, q)^\top$ is the given data and ∇f is known. As boundary condition we may employ $w(0, 0) = 0$. After the computation of w we easily compute the sought depth map v via $v = w - \lambda f$.

The PDE (2.5) is as (2.3) a nonlinear Hamilton-Jacobi equation of which the solution has to be understood in the framework of viscosity solutions [1].

The Concept of Upwinding. In order to obtain a stable method after [5], we employ the upwind discretisation as proposed in [18] of the partial derivatives f, reading as

$$f_x := \Big[\max\Big(\frac{f_{i,j} - f_{i-1,j}}{\Delta x}, \frac{f_{i,j} - f_{i+1,j}}{\Delta x}, 0 \Big) \Big]^2, \tag{2.6}$$

and analogously for f_y, where we make use of grid widths Δx and Δy. Making use of the same discretisation for the components of ∇w, we obtain a quadratic equation one needs to solve in every grid point except at a boundary point at $(0, 0)$. For use of the integrator over non-convex domains see the description in [5]. We explore that the solution can be computed efficiently by the FM method.

The Fast Marching Method. The idea of the FM method may be traced back to Tsitsiklis [23] who solved a Hamilton-Jacobi equation associated to a particular minimum time problem. About the same time, the FM method in its standard format as we employ it here was presented by Sethian [20] and Helmsen *et al.* [8].

Let us briefly describe the FM strategy. The principle behind FM is *causality*, meaning that the information advances from smaller values w to larger values w in one pass, meeting each grid point just once. To this end, one may employ three disjoint sets of nodes as discussed in detail in [21]: $(s1)$ accepted nodes, $(s2)$ trial nodes and $(s3)$ far nodes. For the members of set $(s1)$ the arrival time at $w_{i,j}$ is known and will not be changed. A member $w_{i,j}$ in set $(s2)$ is a neighbour of an accepted node. This is the set where the computation actually takes place and the values of $w_{i,j}$ can still change. In set $(s3)$ are the nodes $w_{i,j}$ where an approximate solution has not yet been computed as these are not in a neighbourhood of a member of (1). The FM algorithm can also be described as follows until all nodes are accepted:

(a) Find the grid point A in $(s2)$ with smallest arrival time and change it to $(s1)$.
(b) Place all neighbors of A into $(s2)$ if they are not there already and compute the arrival time for all them, if they are not already in $(s1)$.
(c) If the set $(s2)$ is not empty, return to (a).

Let us finally note that for initialisation, one may take the node at $(0, 0)$ bearing the boundary condition into set $(s1)$. The key to an efficient implementation is to store the nodes in $(s2)$ in a heap data structure, so that the smallest element in step (a) can be chosen as fast as possible.

2.2. Krylov-based Poisson Integration. Let us briefly review the classic variational approach to this problem cf. [3, 7, 11, 17, 22]. In order to recover the surface it is common to minimise the least squares error between the input and the gradient field of v via minimising

$$J(v) = \iint_\Omega \|\nabla v - \mathbf{g}\|^2 \, dx \, dy = \iint_\Omega (v_x - p)^2 + (v_y - q)^2 \, dx \, dy. \tag{2.7}$$

A minimizer v of (2.7) must satisfy the associated Euler-Lagrange equation which leads to the following *Poisson equation*

$$\Delta v = \operatorname{div}(p, q) = p_x + q_y \tag{2.8}$$

complemented by natural Neumann boundary conditions $(\nabla v - \mathbf{g}) \cdot \mu = 0$, where the vector μ is an outer normal to $\partial\Omega$. Let us note that the handling of strong noisy data and outliers leads to modifications and regularizations with regard to the energy functional in (2.7), see [17] for more details.

Discretisation of the Poisson Equation. A standard numerical approach to solve the Poisson equation in (2.8) makes use of finite differences. Often $\Delta v = v_{xx} + v_{yy}$ and $\operatorname{div}(p, q)$ are approximated by central differences. For simplicity we suppose that the grid size is $\Delta x = \Delta y = 1$. Then a suitable discrete version of (2.8) is given by

$$-4v_{i,j} + (v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1}) = \frac{p_{i+1,j} - p_{i-1,j} + q_{i,j+1} - q_{i,j-1}}{2} \tag{2.9}$$

which corresponds to a linear system $A\mathbf{x} = \mathbf{b}$, where the vectors $\mathbf{x}$ and $\mathbf{b}$ are obtained by stacking the unknown values $v_{i,j}$ and the given data $(p_{i,j}, q_{i,j})$, respectively. The matrix A contains the coefficients arising by the discretisation of the Laplace operator Δ. With respect to the Neumann boundary conditions the matrix A is a rank-1 deficient matrix which is symmetric, positive semidefinite and diagonal dominant.

Preconditioned Krylov Subspace Solvers. A particular class of iterative solvers designed for use with large sparse linear systems such as given by (2.9) is the class of *Krylov subspace solvers*; for a detailed exposition see [19]. The main idea behind the Krylov approach is to search for an approximative solution of $A\mathbf{x} = \mathbf{b}$, with $A \in \mathbb{R}^{n \times n}$ a large regular sparse matrix and $\mathbf{b} \in \mathbb{R}^n$, in a suitable low-dimensional subspace $\mathbb{R}^k$ of $\mathbb{R}^n$ that is constructed iteratively with k being the number of iterates. The aim in the construction of a Krylov subspace method is thereby to have a good representation of the solution after a few iterates.

Let us note that this construction is often not directly visible in the formulation of a Krylov subspace method, as these are often described in terms of a reformulation of the original system $A\mathbf{x} = \mathbf{b}$ as an optimisation task. An important classic example for a Krylov subspace method is the well-known conjugate gradient (CG) method of Hestenes and Stiefel [9]. We will also consider other Krylov subspace methods as mentioned in the experimental section.

It is important to mention that the use of a preconditioner can enhance the performance of a Krylov subspace method significantly. The idea of preconditioning is to multiply the system $A\mathbf{x} = \mathbf{b}$ with a matrix M such that M approximates A^{-1}. The modified system $MA\mathbf{x} = M\mathbf{b}$ is much more efficient to solve as the convergence speed of an iterative solver becomes better as the system matrix, now MA, approximates the identity matrix. In case of sparse matrices A, typically preconditioners are defined over the same sparse structure as of A, and we consider in this work the so-called incomplete LU (ILU) factorisation that represents a sparse version of the classic LU

factorisation; see once more [19] for details. Let us also note that this preconditioner in combination with the BiCGStab or TFQMR method is known to be a highly efficient solver in the context of computational fluid dynamics problems [15]. Furthermore regarding to the CG method, we consider the so-called CMG preconditioner designed by Koutis *et al.* [14].

3. Numerical experiments. Let us now demonstrate the efficiency of the proposed FM-Krylov approach compared to the state of the art. Thereby we give a careful evaluation of all the components of our novel algorithm. On the technical side, let us note that the experiments were conducted on a I7 processor at $2.9GHz$. Besides the quality of results, we also evaluate the considered integration methods with respect to CPU time and also memory requirements as this is an interesting point of the discussion in the recent work [7].

3.1. Fast Approximate Reconstruction. We first consider important properties of the FM integrator and compare with results of the Poisson integration. Let us note that for experiments with the FM integrator we have to define the parameter λ, which is not a crucial choice as discussed in [5]. Any large number $\lambda \gg 0$ will work.

Tested Datasets. To evaluate the efficiency of the proposed algorithm, we consider the datasets presented in Figure 3.1, representing examples of applications of our method to gradient-domain image reconstruction (PET imaging, Poisson image editing) and surface-from-gradient (photometric stereo). The gradients of the Phantom and Lena images were constructed using finite differences, while that of the Beethoven surface was estimated by classic PS [24]. Finally, both the surface and the gradient of the Peaks datasets are analytically known, preventing any bias due to finite difference approximations.

Let us note that our test datasets also address fundamental aspects one may typically find in gradient fields obtained from real-world problems: sharp gradients (Phantom), highly fluctuating gradients oriented in all grid directions representing for instance textured areas (Lena) and smoothly varying gradient fields (Peaks). The gradient field corresponding to the PS solution of the Beethoven bust features difficulties like e.g. self-occlusions at the nose as well as a non-trivial computational domain.

Comparing FM integration with Poisson solvers. The reconstruction results of the FM integrator are demonstrated in the left columns of Figures 3.2, 3.3, and 3.4, and compared to Poisson reconstruction (right column). Both reconstructions are qualitatively quite close. This shows that the FM integrator can be used to get efficiently a first approximation of the Poisson solution. To numerically solve the arising discrete Poisson equations, as we cannot use fast Poisson solvers as e.g. in [22] due to the non-rectangular nature of the domains, we constructed explicitely the linear systems and solved them using a direct solver (Matlab's backslash) as often done by practitioners. This requires a lot more time and memory than the FM integrator.

Robustness. One can observe a certain lack in robustness of the FM integrator, especially in directions not aligned with the grid structure. However, this is a property that is evident because of the causality concept behind the FM scheme; errors that once appear are transported over the computational domain. This is not the case using Poisson reconstruction, which is a global approach and includes a regularising mechanism via the underlying least squares model.

Hence, the FM solver we propose is more efficient than previous approaches in terms of system requirements. Yet, it is quantitatively less accurate and less robust to noise: such an issue can be dealt with by coupling both approaches, considering the

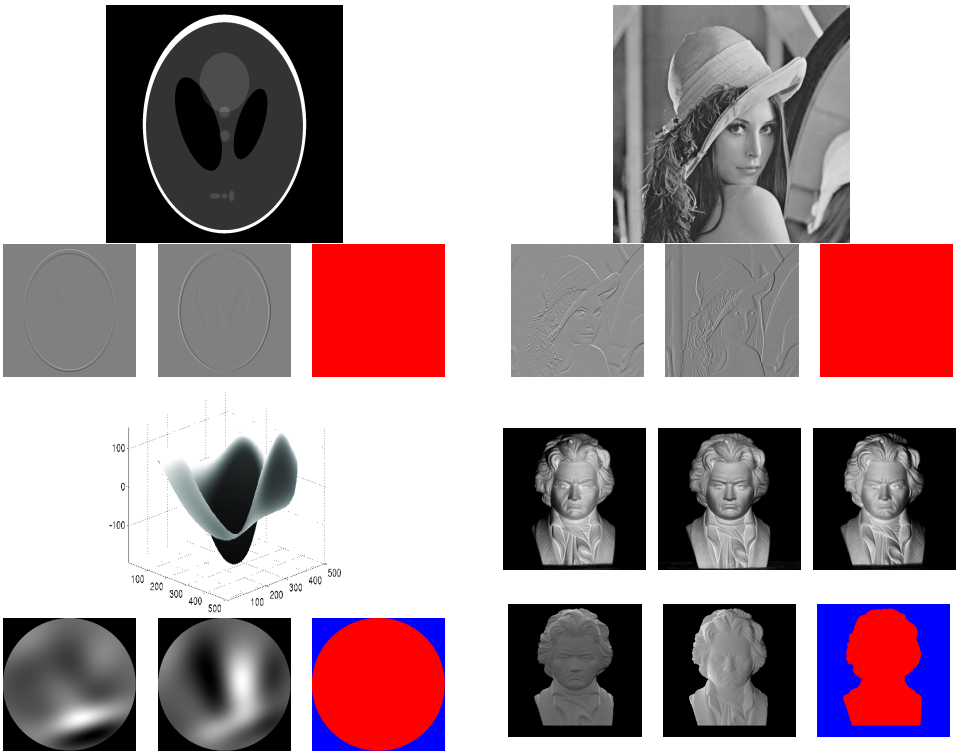


FIG. 3.1. *Data used in the experiments: Phantom (*1024×1024*), Lena (*512×512*), Peaks (*512×512*) and Beethoven (*256×256*) datasets. For each dataset, the top row illustrates the dataset, and the bottom row shows the input data provided to our algorithm (estimation of the gradient in the $x-$ and $y-$ direction, and reconstruction domain). For both first sets, the gradient is estimated by finite differences of the input image, while it is analytically known for Peaks, and estimated by photometric stereo for Beethoven, from the three images shown in the corresponding top row.*

FM reconstruction as an initialization for iterative Poisson solvers based on Krylov subspaces.

3.2. Accelerating Krylov-based Integration. The direct approach to Poisson equation resolution is very memory expensive (it requires almost 1GB of memory for a 1024×1024 dataset, while the FM requires only 115MB). Given the considerations above on the accuracy of FM results, we propose to use FM as an initial provider for the depth map, which is further refined using Krylov solvers. This is illustrated in Figure 3.5, where it is shown that choosing the FM as initialization improves a lot the convergence rate of the LSQR method for solving the discrete Poisson equation.

Coupling appropriate initialization with efficient preconditioning. It has been experimentally shown in [7] that the LSQR method, which is a classic method proposed for such a task [16], is inefficient at solving the integration problem. This serves as an argument in [7] to justify a faster Sylvester-based approach, rather than the traditional Poisson one. Yet, this approach requires the reconstruction domain to be rectangular, a strong assumption which is rarely desirable in surface reconstruction. We show in Figure 3.6 that there are much faster Krylov solvers for solving the problem: replacing the solver from LSQR to CG reduces the iteration number required to get a 10^{-2} relative residual from 10000 to only 400. Using an appropriate precondi-

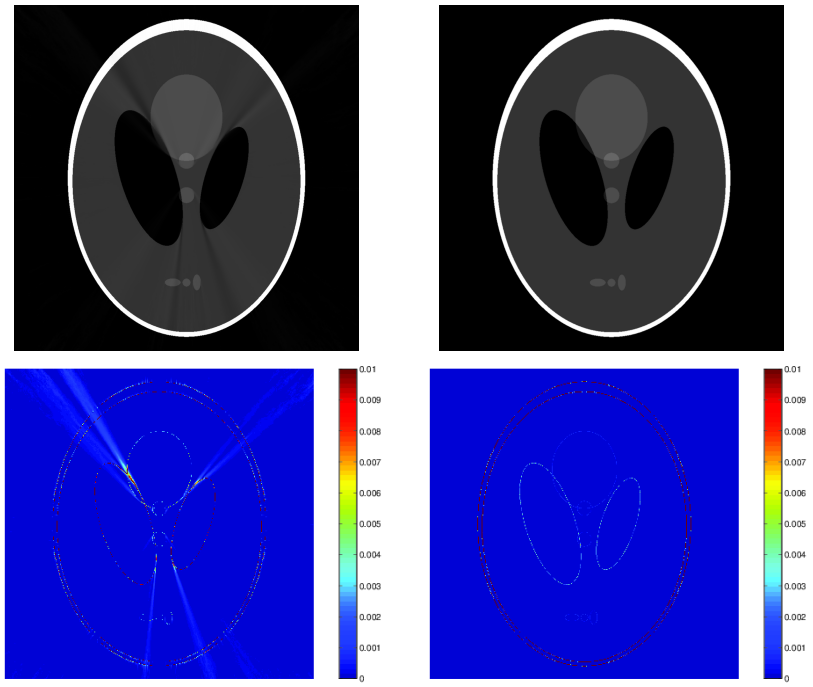


FIG. 3.2. *Results on the* 1024×1024 *'Phantom' dataset. Left: FM result; Right: Poisson result. Top row: reconstruction result. Bottom row: squared error map. The two results are roughly equivalent, but the FM approach is much faster and less memory-hungry. On the other hand, it produces radial artifacts and is thus qualitatively less accurate. Such artifacts are easily smoothed when applying iterative Poisson resolution afterwards.*

tioner such as the mentioned CMG scheme, the relative residual at just a quarter of these iterations is even reduced from to 10^{-2} to below 10^{-8}, cf. Figure 3.6.

Similar tests were conducted on the following solvers (considering the ILU preconditioning): BICGSTAB, BICGSTAB(l), CGS, GMRES, LSQR, MINRES, QMR and TFQMR, cf. [19]. According to our experiments, CG and TFQMR are superior, whose convergence rates are illustrated in Figure 3.6. These experiments show that, by choosing appropriate initialization (FM) and preconditioner (CMG or ILU), the iterative FM-Krylov methods for solving the discrete Poisson equation provide fast and accurate 3D-reconstruction, while requiring limited memory and making no assumption on the shape of the reconstruction domain. We emphasize that this is a relevant step forward with respect to the literature, where all these features were never simultaneously demonstrated.

4. Conclusion. We demonstrated the properties of our novel FM-Krylov surface normal integrator. It combines all efficiency benefits of the FM and Krylov-based components while retaining the robustness and accuracy of the underlying variational approach. Recalling the properties of an ideal surface normal integrator as discussed in the introduction, we think that all the desirable points including especially the flexibility in handling non-trivial domains are met by the proposed method.

REFERENCES

[1] G. Barles. *Solutions de viscosité des équations de Hamilton Jacobi*. Springer, 1994.

[2] J. D. Durou, J. F. Aujol, and F. Courteille. Integrating the normal field of a surface in the presence of discontinuities. *Energy Minimization Methods in Computer Vision and Pattern Recognition*, 5681:261–273, 2009.

[3] J. D. Durou and F. Courteille. Integration of a normal field without boundary condition. *Proc. of the First International Workshop on Photometric Analysis For Computer Vision*, 2007.

[4] R. T. Frankot and R. Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 10(4):439–451, 1988.

[5] S. Galliani, M. Breuß, and Y. C. Ju. Fast and robust surface normal integration by a discrete eikonal equation. *Proc. British Machine Vision Conference*, pages 1–11, 2012.

[6] M. Harker and P. O' Leary. Least squares surface reconstruction from measured gradient fields. *In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[7] M. Harker and P. O'Leary. Regularized reconstruction of a surface from its measured gradient field. *Journal of Mathematical Imaging and Vision*, 51(1):46–70, 2015.

[8] J. J. Helmsen, E. G. Puckett, P. Colella, and M. Dorr. Two new methods for simulating photolithography development in 3d. *Optical Microlithography IX*, 2726:253–261, 1996.

[9] M.R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *NBS Journal of Research*, 49:409–436, 1952.

[10] J. Ho, J. Lim, M. H. Yang, and D. Kriegmann. Integrating surface normal vectors using fast marching method. *In Proc. European Conference on Computer Vision*, 3953:239–250, 2006.

[11] B. K. P. Horn and M. J. Brooks. The variational approach to shape from shading. *Computer Vision, Graphics and Image Processing*, 33(2):174–208, 1986.

[12] I. Horowitz and N. Kiryati. Depth from gradient fields and control points: bias correction in photometric stereo. *Image and Vision Computing*, 22:681–694, 2004.

[13] R. Klette and K. Schlüns. Height data from gradient fields. *Proc. International Society for Optical Engineering*, 2908:204–215, 1996.

[14] I. Koutis, G. L. Miller, and R. Peng. A Nearly-m log n Time Solver for SDD Linear Systems. In *IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 590–598, Palm Springs, États-Unis, 2011.

[15] A. Meister. Comparison of different Krylov subspace methods embedded in an implicit finite volume scheme for the computation of viscous and inviscid flow fields on unstructured grids. *Journal of Computational Physics*, 140:311–345, 1998.

[16] C. Paige and M. Saunders. Lsqr: An algorithm for sparse linear equations and sparse least-squares. *ACM Transactions on Mathematical Software*, 8(1):43–71, 1982.

[17] Y. Quéau and J.-D. Durou. Edge-preserving integration of a normal field: Weighted least squares, tv and l1 approaches. In J. F. Aujol, M. Nikolova, and N. Papadakis, editors, *Scale Space and Variational Methods in Computer Vision*, volume 2749 of *Lecture Notes in Computer Science*, pages 576–588. Springer International Publishing, 2015.

[18] E. Rouy and A. Tourin. A viscosity solutions approach to shape-from-shading. *SIAM Journal on Numerical Analysis*, 29(3):867–884, 1992.

[19] Y. Saad. *Iterative Methods For Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2nd Edition, 2003.

[20] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. of the National Academy of Sciences of the United States of America*, 93(4):1591–1595, 1996.

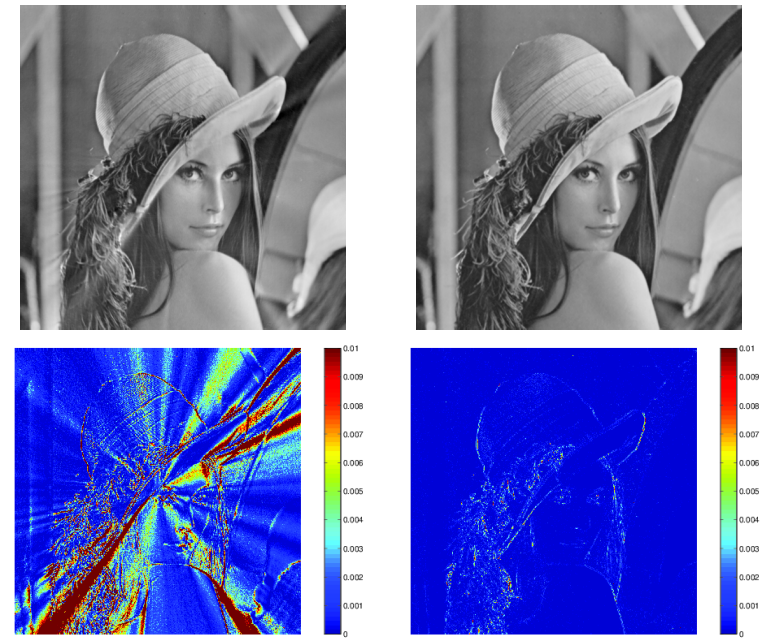
[21] J. A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 2nd Edition, 1996.

[22] T. Simchony, R. Chellappa, and M. Shao. Direct analytical methods for solving Poisson equations in computer vision problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):435–446, 1990.

[23] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

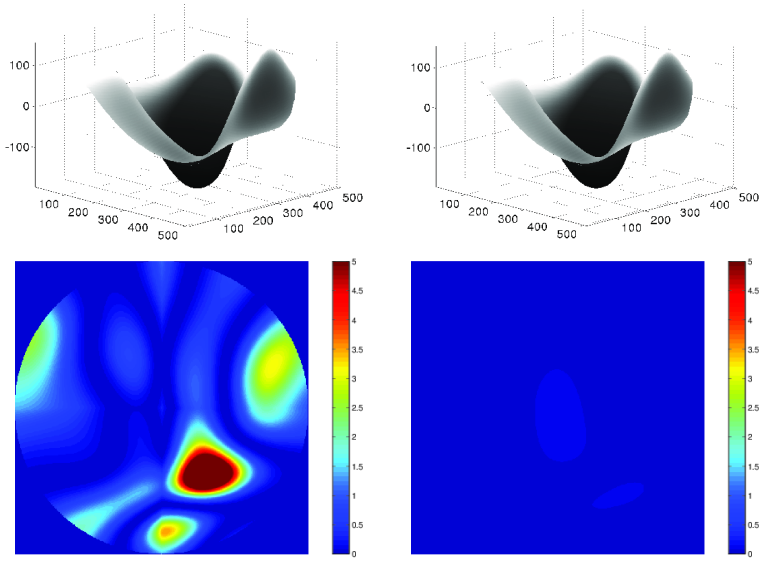
[24] R.J̃. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):134–144, 1980.

[25] Z. Wu and L. Li. A line-integration based method for depth recovery from surface normals. *Computer Vision, Graphics and Image Processing*, 43(1):53–66, 1988.



MSE $= 35 \times 10^{-4}$
CPU $= 0.27s$
Mem. Peak $= 40$MB

MSE $= 1.8 \times 10^{-4}$
CPU $= 1.8s$
Mem. Peak $= 180$MB



MSE $= 0.91$
CPU $= 0.28s$
Mem. Peak $= 40$MB

MSE $= 0.02$
CPU $= 1.3s$
Mem. Peak $= 160$MB

FIG. 3.3. *Results on the* 512×512 *'Lena' and 'Peaks' datasets. Though the squared error map demonstrates the artifacts due to the propagating nature of the FM scheme, the reconstruction result is qualitatively very satisfactory, while requiring much less resources, compared to Poisson solvers (right column).*

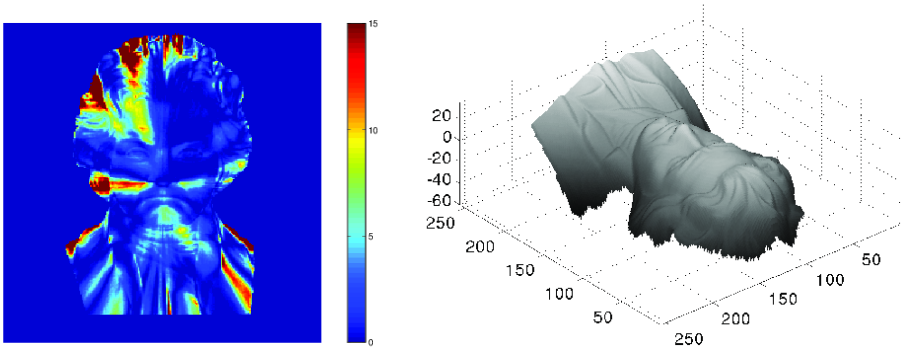


FIG. 3.4. *Results on the* 256×256 *'Beethoven' dataset. Left: squared error map of FM result; Right: reconstruction result of FM. The results of the FM scheme are qualitatively similar to the solution by Poisson reconstruction, though it as a much lower complexity. The left figure shows the difference from FM to the Poisson reconstruction, putting the latter into the role of the ground truth because of its overall higher accuracy.*

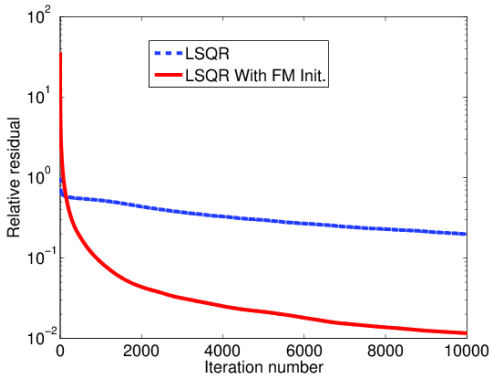


FIG. 3.5. *Convergence rate of the LSQR method, considering flat vs. FM initialisation. We show the relative residual as a function of the iteration. Convergence is accelerated a lot using the FM initialisation, though LSQR is not efficient for this problem (see Figure 3.6).*

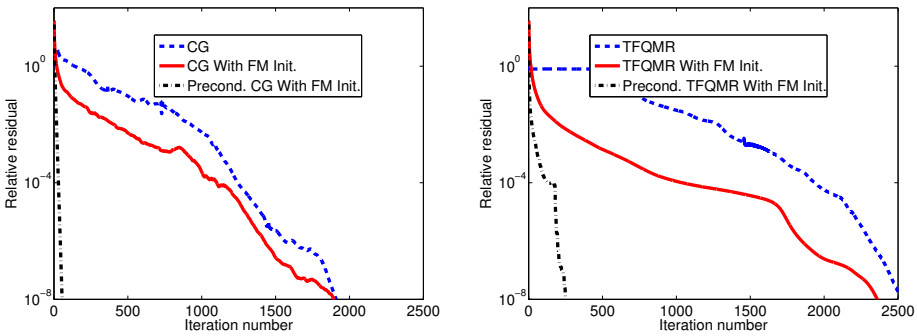


FIG. 3.6. *Convergence rates for CG and TFQMR, for flat or FM initialization, and with or without preconditioning (CMG for CG and ILU for TFQMR). Coupling FM initialisation with appropriate preconditioning provides enhanced convergence rates.*