Computational Visual Media
DOI 10.1007/s41095-016-0075-z Vol. 3, No. 2, June 2017, 107–129

Research Article

Fast and accurate surface normal integration on non-rectangular domains

Martin Bähr[1], Michael Breuß[1] (✉), Yvain Quéau[2], Ali Sharifi Boroujerdi[1], and Jean-Denis Durou[3]

Abstract The integration of surface normals for the purpose of computing the shape of a surface in 3D space is a classic problem in computer vision. However, even nowadays it is still a challenging task to devise a method that is flexible enough to work on non-trivial computational domains with high accuracy, robustness, and computational efficiency. By uniting a classic approach for surface normal integration with modern computational techniques, we construct a solver that fulfils these requirements. Building upon the Poisson integration model, we use an iterative Krylov subspace solver as a core step in tackling the task. While such a method can be very efficient, it may only show its full potential when combined with suitable numerical preconditioning and problem-specific initialisation. We perform a thorough numerical study in order to identify an appropriate preconditioner for this purpose. To provide suitable initialisation, we compute this initial state using a recently developed fast marching integrator. Detailed numerical experiments illustrate the benefits of this novel combination. In addition, we show on real-world photometric stereo datasets that the developed numerical framework is flexible enough to tackle modern computer vision applications.

Keywords surface normal integration; Poisson integration; conjugate gradient method; preconditioning; fast marching method; Krylov subspace methods; photometric stereo; 3D reconstruction

1 Introduction

The integration of surface normals is a fundamental task in computer vision. Classic examples of processes where this technique is often applied are image editing [1], shape from shading as analysed by Horn [2], and photometric stereo (PS) for which we refer to the pioneering work of Woodham [3]. Modern applications of PS include facial recognition [4], industrial product quality control [5], object preservation in digital heritage [6], and new utilities with potential use in the video (game) industry and robotics [7], among many others.

In this paper we consider surface normal integration in the context of the PS problem, which serves as a role model for potential applications. The task of PS is to compute the 3D surface of an object from multiple images of the same scene under different illumination conditions. The standard PS method for reconstructing an unknown surface has two stages. In a first step, the surface is represented as a field of surface normals, or equivalently, a corresponding gradient field. In a subsequent step this is integrated to obtain the depth of the surface.

To handle the integration step, many different approaches and methods have been developed during recent decades. However, despite all these developments there is still the need for approaches that combine a high *accuracy* reconstruction with *robustness* against noise and outliers, and reasonable computational *efficiency* for working with high-

1 Brandenburg Technical University, Institute for Mathematics, Chair for Applied Mathematics, Platz der Deutschen Einheit 1, 03046 Cottbus, Germany. E-mail: M. Bähr, martin.baehr@b-tu.de; M. Breuß, michael.breuss@b-tu.de (✉); A. S. Boroujerdi, ali.sharifiboroujerdi@b-tu.de.

2 Technical University Munich, 85748 Garching, Germany. E-mail: yvain.queau@tum.de.

3 Université de Toulouse, IRIT, UMR CNRS 5505, Toulouse, France. E-mail: durou@irit.fr.

Manuscript received: 2016-10-18; accepted: 2016-12-21

resolution cameras and corresponding imagery.

Computational issues. Let us briefly elaborate on the demands on an ideal integrator. As discussed, e.g., in Ref. [8], a practical issue is robustness with respect to noise and outliers. Since computer vision processes such as PS rely on simplified assumptions that often do not hold for realistic illumination and surface reflectance, artefacts may often arise when estimating surface normals from real-world input images. Therefore, the determined depth gradient field is not noise-free and may also contain outliers. These may strongly influence the integration process.

Secondly, objects to be reconstructed in 3D are typically in the centre of a photographed scene. Therefore, they only form part of each input image. The sharp gradient usually representing the transition from foreground to background is a difficult feature for most surface normal integrators and generally influences the estimated shape of the object of interest. Because of this it is desirable to consider only image segments that represent the object of interest and not the background. Although similar difficulties (sharp gradients) also arise for discontinuous surfaces that may appear at self-occlusions of an object [9], we do not tackle this issue in the present article. Instead, we neglect self-occlusion and focus on reconstructing a *smooth* surface on the (possibly non-rectangular) subset of the image domain representing the object of interest. Related to this point, another important aspect is the computational time and cost saving that can be achieved by the concomitant decrease in the number of elements of the computational domain. For reasons of both reconstruction quality and efficiency, an ideal solver for surface normal integration should thus work on non-rectangular domains.

Finally, to capture ever more detail in 3D reconstruction, camera technology evolves and the resolution of images tends to increase continually. This means, the integrator has to work accurately and quickly for various sizes of input images, including images of size at least 1000×1000 pixels. Consequently, the computational efficiency of a solver is a key requirement for many possible modern and future applications.

To summarise, one can identify the desirable properties of *robustness* with respect to noise and outliers, ability to work on *non-rectangular domains*, and *efficiency* of the method: one aims for an accurate solution using reasonable computational resources (such as time and memory).

Related work. Many methods taking into account the abovementioned issues individually have been developed to solve the problem of surface normal integration during recent decades.

According to Klette and Schlüns [10], such methods can be classified as local and global integration methods.

The most basic local method, also referred to as direct line-integration scheme [11–13], is based on the line integral technique and the fact that a closed path on a continuous surface should be zero. These methods are in general quite fast, but due to their local nature, the reconstructed solution depends on the integration path. Another more recent local approach for normal integration is based on an eikonal-type equation, which can be solved by applying the computationally efficient fast marching (FM) method [14–16]. However, a common disadvantage of all local approaches is sensitivity with regard to noise and discontinuities, which lead to error accumulation in the reconstruction.

In order to minimise error accumulation, it is preferable to adopt a global approach based on the calculus of variations. Horn and Brooks [2] proposed the classic and most natural variational method for the intended task by casting the corresponding functional in a form that results in a least-squares approximation. The necessary optimality condition represented by the Euler–Lagrange equation of the classic functional is given by the Poisson equation, which is an elliptic partial differential equation. This approach to surface normal integration is often called *Poisson integration*. The practical task arising amounts to solving the linear system of equations that corresponds to the discretised Poisson equation.

Direct methods for solving the latter system, as for instance Cholesky factorisation, can be fast, but this type of solver may use a substantial amount of memory and appears to be rather impractical for images of larger than 1000×1000 pixels. Moreover, if based on matrix factorisation, the factorisation itself is relatively expensive to compute. Generally, direct methods offer an extremely highly accurate result, but one must pay a high computational price. In contrast, *iterative methods* are not naturally noted

for extremely high accuracy but are very fast when computing approximate solutions. They require less memory and are thus inherently more attractive candidates for this application, but involve some non-trivial aspects (see later) which make them less straightforward to use.

An alternative approach to solving the least-squares functional was introduced by Frankot and Chellappa [17]. The main idea is to transform the problem to the frequency domain where a solution can be computed in linear time through the fast Fourier transform, if periodic boundary conditions are assumed. The latter unfavourable condition can be resolved by use of the discrete cosine transform (DCT) as shown by Simchony et al. [18]. However, these methods remain limited to rectangular domains. To apply these methods on non-rectangular domains requires introducing zero-padding in the gradient field which may lead to an unwanted bias in the solution. Some conceptually related basis function approaches include the use of wavelets [19] and shapelets [20]. The method of Frankot and Chelappa was enhanced by Wei and Klette [21] to improve its accuracy and robustness to noise. Another approach was proposed by Karaçali and Snyder [22] who make use of additional adaptive smoothing for noise reduction.

Among all of the mentioned global techniques, *variational methods* offer a high robustness with respect to noise and outliers. Therefore, many extensions have been developed in modern works [9, 23–28]. Agrawal et al. [23] use anisotropic instead of isotropic weights for the gradients during integration. Durou et al. [9] give a numerical study of several functionals, in particular with non-quadratic and non-convex regularisations. To reduce the influence of outliers, the L_1 norm has also become an important regularisation instrument [25, 26]. In Ref. [24] the extension to L_p minimisation with $0 < p < 1$ is presented. Two other recent works are Ref. [27] where the use of alternative optimisation schemes is explored and Ref. [28] where the proposed formulation leads to the task of solving a Sylvester equation. Nevertheless, these methods have some drawbacks. By the application of additional regularisation as in Refs. [9, 23–27], depth reconstruction becomes quite time-consuming and the correct setting of parameters is more difficult, while the approach of Harker and O'Leary [28] is only efficient for rectangular domains Ω.

Summarizing the achievements of previous works, the problem of surface normal integration on non-rectangular domains has not yet been perfectly solved. The main challenge is still to find a balance between quality and time needed to generate the result. One should take into account that to achieve better quality in 3D object reconstruction, the resolution of images tends to increase continually, and so computational efficiency is surely a key requirement for many potential current and future applications.

Our contributions. To balance the aspects of quality, robustness, and computational efficiency, we go back to the powerful classic approach of Horn and Brooks as the variational framework has the benefit of high modeling flexibility. In detail, our contributions when extending this classic path of research are:

1. Building upon a recent conference paper where we compared several Krylov subspace methods for surface normal integration [29], we investigate the use of the preconditioned conjugate gradient (PCG) method for performing Poisson integration over non-trivial computational domains. While such methods constitute advanced yet standard methods in numerical computing [30, 31], they are not yet standard tools in image processing, computer vision, and graphics. To be more precise, we propose to employ the conjugate gradient (CG) scheme as the iterative solver and we explore modern variations of incomplete Cholesky (IC) decomposition for preconditioning. The thorough numerical investigation here represents a significant extension of our conference paper.
2. For computing a good initialisation for the PCG solver, we employ a recent FM integrator [15] already mentioned above. Its main advantages are its flexibility for use with non-trivial domains coupled with low computational requirements. While we proposed this means of initialisation already in Ref. [29], our numerical extensions mean that the conclusion we draw in this paper is much sharper.
3. We prove experimentally that our resulting, combined novel method unites the advantages of

flexibility and *robustness* of variational methods with *low computational time* and *low memory requirements*.

4. We propose a simple yet effective modification for gradient fields containing severe outliers, for use with Poisson integration methods.

The abovementioned building blocks of our method represent a pragmatic choice among current tools for numerical computing. Moreover, as demonstrated by our new integration model that is specifically designed for tackling data with outliers, our numerical approach can be readily adapted to other Poisson-based integration models. This, together with the well-engineered algorithm for our application, i.e., FM initialisation and fine-tuned algorithmic parameters, makes our method a unique, efficient, and flexible procedure.

2 Surface normal integration

The mathematical set-up of surface normal integration (SNI) can be described as follows. We assume that for a domain Ω, a normal field $\boldsymbol{n} := \boldsymbol{n}(x,y) = [n_1(x,y), n_2(x,y), n_3(x,y)]^{\mathrm{T}}$ is given for 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 $\boldsymbol{n}$ is the normal field of v. Assuming orthographic projection①, the normal field $\boldsymbol{n}$ of a surface at $(x,y,v(x,y)) \in \mathbb{R}^3$ can be written as

$$\boldsymbol{n}(x,y) := \frac{[-v_x, -v_y, 1]^{\mathrm{T}}}{\sqrt{\|\nabla v\|^2 + 1}} \tag{1}$$

with $v_x := \partial v/\partial x$, $v_y := \partial v/\partial y$, and $\nabla v := [v_x, v_y]^{\mathrm{T}}$. Moreover, the components of $\boldsymbol{n}$ are given by partial derivatives of v:

$$(v_x, v_y) = \left(-\frac{n_1}{n_3}, -\frac{n_2}{n_3}\right) = (p,q) \tag{2}$$

where we think of p and q as given data.

In this section, we present the building blocks of our new algorithm in two steps, first the fast marching integrator, and afterwards the iterative Poisson solver relying on the conjugate gradient method supplemented by (modified) incomplete Cholesky preconditioning. When presenting Poisson integration, we also demonstrate the flexibility of the resulting discrete computational model by a novel adaptation for handling data with outliers.

① The perspective integration problem can be formulated in a similar way, using the change of variable $v = \log v$ [32].

A detailed description of the fast marching integrator can be found in Refs. [15, 16], and the presentation of the components of the CG scheme can be found in literature on Krylov subspace solvers, e.g., Ref. [31]. We still summarize the algorithms in some detail here because there are important parameters that need to be set and some choices to make: since the efficiency of integrators depends largely on such practical implementation details, our explanations provide additional value beyond a plain description of the methods.

While our discretisation of the Poisson equation is a standard one, we deal with non-trivial boundary conditions in our application, necessitating a thorough description. The construction of our non-standard numerical boundary conditions, which is often overlooked in the literature, is another technical contribution to the field.

2.1 Fast marching integrator

We recall for the convenience of the reader some relevant developments from Refs. [14–16], which showed that it is possible to tackle the problem of surface normal integration via the following PDE-based model in $w = v + \lambda f$:

$$\|\nabla w\| = \sqrt{(p + \lambda f_x)^2 + (q + \lambda f_y)^2} \tag{3}$$

where $\lambda > 0$ and $f : \mathbb{R}^2 \to \mathbb{R}$ are user-defined. Using PDE (3) we do not compute the depth function v directly, but instead we solve in a first step for a function w. This means, to obtain v one has to solve the eikonal-type equation for w, in which $\nabla v = (p,q)$ and ∇f are known, and recover v in a second step from the computed w by subtracting the known function f. The intermediate step of considering a new function w is necessary for successful application of the FM method, in order to avoid local minima and ensure that any initial point can be considered [14]. It turns out that a natural candidate for f is the squared Euclidean distance function with its minimum in the centre of the domain $(x_0, y_0) = (0,0)$, i.e.,

$$f := f(x,y) = x^2 + y^2 \tag{4}$$

Other choices for f are also possible [16]. As boundary condition we may employ $w(0,0) = 0$. After computation of w we easily compute the sought depth map v via $v = w - \lambda f$. Let us note that the FM integrator requires parameter λ to be tuned, but it is not a crucial choice as any large number $\lambda \gg 0$

will work [15]①.

Numerical upwinding. A crucial issue for the FM integrator is correct discretisation of the derivatives of f in Eq. (3). In order to obtain a stable method, an upwind discretisation of the partial derivatives of f is required:

$$f_x := \Big[\max\Big(\frac{f_{i,j}-f_{i-1,j}}{\Delta x}, \frac{f_{i,j}-f_{i+1,j}}{\Delta x}, 0\Big)\Big]^2 \quad (5)$$

and analogously for f_y, for grid widths Δx and Δy.

Making use of the same discretisation for the components of ∇w, one obtains a quadratic equation that must be solved for every pixel except at the initial pixel $(0,0)$ where some depth value is prescribed.

Let us note that the initial point can be chosen in practice anywhere, i.e., there is no restriction to $(0,0)$.

Non-convex domains. If the above method is used without modification over non-convex domains, the FM integrator fails to reconstruct the solution. The reason is that the original, unmodified squared Euclidean distance does not yield a meaningful distance from the starting point to pixels which are not connected by a direct line lying within the integration domain. In other words, the unmodified scheme just works over convex domains.

To overcome the problem, a suitable distance which calculates the shortest path from the starting point to every point on the computational domain is necessary. To this end, the use of a geodesic distance function d is advocated [15]. We proceed as follows, relying on similar ideas to those in, e.g., Ref. [33] for path planning. In a first step we solve an eikonal equation $\|\nabla d\| = 1$ over all the points of the domain with $d := 0$ at the chosen start point. This can of course be done again with the FM method. Then, in a second step we are able to compute the depth map v. Therefore, we use Eq. (3) for w, with the squared geodesic distance function d instead of f and using Eq. (5). Afterwards we recover v via $v = w - \lambda d$.

Fast marching algorithm. The idea of FM goes back to Refs. [34–36]. For a comprehensive introduction see Ref. [37]. The benefit of FM is its relatively low complexity of $O(n \log n)$ where n is the number of points in the computational domain②.

Let us briefly describe the FM strategy. The principle behind FM is that information advances from smaller values of w to larger values of w, thus visiting each point of the computational domain just once. To this end, one may employ three disjoint sets of nodes as discussed in detail in Refs. [37, 39]: $\{s_1\}$ accepted nodes, $\{s_2\}$ trial nodes, and $\{s_3\}$ far nodes. The values $w_{i,j}$ for set $\{s_1\}$ are considered known and will not be changed. A member $w_{i,j}$ in set $\{s_2\}$ is always at 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. Set $\{s_3\}$ contains those nodes $w_{i,j}$ where an approximate solution has not yet been computed as these are not in a neighbourhood of a member of $\{s_1\}$.

The FM algorithm iterates the following procedure until all nodes are accepted:

(a) Find the grid point A in $\{s_2\}$ with the smallest value and move it to $\{s_1\}$.

(b) Place all neighbours of A into $\{s_2\}$ if not already there and compute the arrival time for all of them, if they are not already in $\{s_1\}$.

(c) If the set $\{s_2\}$ is not empty, return to (a).

For initialisation, one may start by putting the node at $(0,0)$ into set $\{s_1\}$; it bears the boundary condition of the PDE (3).

An efficient implementation amounts to storing the nodes in $\{s_2\}$ in a heap data structure, so the smallest element in step (a) can be chosen as quickly as possible.

2.2 Poisson integration

The first part of this section is dedicated to *modeling*. We first briefly review the classic variational approach to the Poisson integration (PI) problem [2, 18, 27, 28, 32]. The handling of extremely noisy data motivates modifications of the underlying energy functional (6), e.g., see Ref. [27]. By proposing a new model dealing with outliers, we demonstrate that the Poisson integration framework is flexible enough to deal with such modern approaches.

The second part is devoted to the *numerics*. We propose a dedicated, and somewhat non-standard, discretisation for our application.

Classic Poisson integration model. In order to recover the surface it is common to minimise the least-squares error between the input and the gradient field of v by minimising:

$$J(v) = \iint_\Omega \|\nabla v - \boldsymbol{g}\|^2 \,\mathrm{d}x\,\mathrm{d}y$$

①In our experiments, we used the value $\lambda = 10^5$.

②When using the *untidy priority queue* structure [38] the complexity may even be lowered to $O(n)$.

$$= \iint_{\Omega} \left[(v_x - p)^2 + (v_y - q)^2 \right] \mathrm{d}x\mathrm{d}y \tag{6}$$

where $\boldsymbol{g} = [p, q]^{\mathrm{T}}$.

A minimiser v of Eq. (6) must satisfy the associated Euler–Lagrange equation which is equivalent to the following *Poisson equation*:

$$\Delta v = \operatorname{div}(p, q) = p_x + q_y \tag{7}$$

that is usually complemented by (natural) Neumann boundary conditions $(\nabla v - \boldsymbol{g}) \cdot \boldsymbol{\mu} = 0$, where the vector $\boldsymbol{\mu}$ is normal to $\partial\Omega$. In this case, uniqueness of the solution is guaranteed, apart from an additional constant. Thus, one recovers the shape but not absolute depth (as in FM integration).

A modified PDE for normal fields with outliers. We now demonstrate by giving an example that the PI framework is flexible enough to also deal with gradient fields featuring strong outliers. To this end, we propose a simple, yet effective way to modify the PI model in order to limit the influence of outliers. Other variations for different applications, e.g., self-occlusions [9, 27], are of course also possible.

Let us briefly recall that the classic model in Eq. (6) which leads to the Poisson equation in Eq. (7) is based on a simple least-squares approach. At locations (x, y) corresponding to outliers, the values $p(x, y)$ and $q(x, y)$ are not reliable, and one would prefer to limit the influence of such corrupt data.

Therefore, we modify the Poisson equation in Eq. (7) by introducing a space-dependent fidelity term $\nu := \nu(x, y)$ by

$$\Delta v = \nabla \cdot \left(\frac{1}{1 + \nu} . [p, q]^{\mathrm{T}} \right) \tag{8}$$

Let us note that a similar strategy, namely to introduce modeling improvements in a PDE that is originally the Euler–Lagrange equation of an energy functional, instead of modifying it, is occasionally employed in computer vision: see, e.g., Ref. [40]. However, we do not tinker here with the core of the PDE, i.e., the Laplace operator Δ, but merely include preprocessing by modifying the right hand side of the Poisson equation.

The key to effective preprocessing is of course to consider the role of ν so that it smooths the surface only at locations where the input gradient is unreliable. Thus, we seek a function $\nu(x, y)$ which is close to zero if the input gradient is reliable, and takes high values if it is not.

The *integrability* term

$$\mathcal{I}(x, y) := p_y - q_x = \nabla \cdot [-q, p]^{\mathrm{T}} \tag{9}$$

should vanish if the surface is $\mathcal{C}^2$-smooth. This argument was used in Ref. [27] to suggest an integrability-based weighted least-squares functional able to recover discontinuity jumps, which generally correspond to a high absolute value of integrability.

Since integrability not only indicates the location of discontinuities, but also that of the outliers, we suggest use of this integrability term to find a smooth surface explaining a corrupted gradient. To do so, we use the following choice for our regularisation parameter:

$$\nu(x, y) = \exp\left(\mathcal{I}(x, y)^2\right) - 1 \tag{10}$$

for which the desired properties (i) vanish when integrability is low (reliable gradients), and (ii) take a high value when integrability is high (outliers).

Putting Eqs. (9) and (10) in Eq. (8), our new model amounts to solving the following equation:

$$\begin{aligned} \Delta v =& \nabla \cdot \left(\frac{1}{1 + \exp\left((p_y - q_x)^2\right) - 1} [p, q]^{\mathrm{T}} \right) \\ =& \nabla \cdot \left[\frac{p}{\exp\left((p_y - q_x)^2\right)}, \frac{q}{\exp\left((p_y - q_x)^2\right)} \right]^{\mathrm{T}} \\ =:& \nabla \cdot [\bar{p}, \bar{q}]^{\mathrm{T}} \end{aligned} \tag{11}$$

which is another Poisson equation, where the right hand side can be computed *a priori* from the input gradient.

Let us clarify explicitly that the meaning of Eq. (11) is to replace the vector of given data $[p, q]^{\mathrm{T}}$ describing the normal field by a modified version $[\bar{p}, \bar{q}]^{\mathrm{T}}$ as defined in Eq. (11).

In addition, we emphasise that *all* methods for SNI based on such a Poisson equation are straightforward to adapt: it suffices to replace (p, q) by $(\bar{p}, \bar{q})$. The algorithmic complexity of all of such approaches remains exactly the same. The practical validity of this simple new model and its benefit of better numerics are demonstrated in Section 4.

In the main part of our paper, for simplicity of presentation, we will simply consider the classic model in Eq. (7) and come back to the proposed modification in Section 4.5.

Discretisation of the Poisson equation. A useful standard numerical approach to solving the Poisson PDE as in Eqs. (7) or (11) makes use of finite differences. Often, $\operatorname{div}(p, q)$ and $\Delta v = v_{xx} + v_{yy}$ are approximated by central differences. For simplicity, we suppose that the grid size is $\Delta x = \Delta y = 1$ as common practice in image processing. Then, a

suitable discrete version of the Laplacian is given in stencil notation by

$$\Delta v(x_i, y_j) \approx \begin{array}{|c|c|c|}\hline & 1 & \\ \hline 1 & -4 & 1 \\ \hline & 1 & \\ \hline\end{array} \cdot\, v_{i,j} \tag{12}$$

so the divergence is given by

$$\mathrm{div}(p_{i,j}, q_{i,j}) \approx \frac{1}{2}\begin{array}{|c|c|c|}\hline \text{-1} & 0 & 1 \\ \hline\end{array} \cdot p_{i,j} + \frac{1}{2}\begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline \text{-1} \\ \hline\end{array} \cdot q_{i,j} \tag{13}$$

where the measured gradient $\boldsymbol{g} = [p, q]^{\mathrm{T}}$. Making use of Eqs. (12) and (13) to discretize Eq. (7) leads to

$$\begin{aligned} &-4v_{i,j} + (v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1}) \\ =&\frac{p_{i+1,j} - p_{i-1,j} + q_{i,j+1} - q_{i,j-1}}{2} \end{aligned} \tag{14}$$

which corresponds to a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where the vectors $\boldsymbol{x}$ and $\boldsymbol{b}$ are obtained by stacking the unknown values $v_{i,j}$ and the given data, respectively. The matrix $\boldsymbol{A}$ contains the coefficients arising by discretizing the Laplace operator Δ.

We employ in all experiments here the above discretisation, as it is very simple and gives high quality results. While other discretisations, e.g., of higher order, are of course possible [28], let us note that this requires one to change the parameter settings we propose for the method. One would also have to adapt the dedicated numerical boundary conditions.

Non-standard numerical boundary conditions. At this point it should be noted that the stencils in Eqs. (12), (13), and the subsequent equation Eq. (14) are only valid for inner points of the computational domain. Indeed, when pixel (i, j) is located near the border of Ω, some of the four neighbour values $\{v_{i+1,j}, v_{i-1,j}, v_{i,j+1}, v_{i,j-1}\}$ in Eq. (14) refer to depths outside Ω. The same holds for the data values $\{p_{i+1,j}, p_{i-1,j}, q_{i,j+1}, q_{i,j-1}\}$: some of these values are unknown when (i, j) is near the border. To handle this, a numerical boundary condition must be invoked.

Using empirical Dirichlet (e.g., using the discrete sine transform [18]) or homogeneous Neumann boundary conditions [23] may result in biased 3D reconstructions near the border. The so-called "natural" condition $(\nabla v - \boldsymbol{g}) \cdot \boldsymbol{\mu} = 0$ [2] is preferred, because it is the only one which is justified.

Let us emphasise that it is not a trivial task to define suitable boundary conditions for $\{p_{i+1,j}, p_{i-1,j}, q_{i,j+1}, q_{i,j-1}\}$. As we opt for a common strategy for discretising values of p, q, v, we employ the following non-standard procedure which has turned out to be preferable in experimental evaluations. Whenever p, q, v values outside Ω are involved in Eq. (14), we discretise this boundary condition using the mean of forward and backward first-order finite differences. This allows us to express the values outside Ω in terms of values inside Ω. To clarify this idea, we distinguish the boundaries according to the number of missing neighbours.

When only one neighbour is missing. There are four types of boundary pixels having exactly one of the four neighbours outside Ω (lower, upper, right, and left borders respectively). Let us first consider the case of a "lower boundary", i.e., a pixel $(i, j) \in \Omega$ such that $(i-1, j), (i+1, j), (i, j+1) \in \Omega^3$ but $(i, j-1) \notin \Omega$. Then, Eq. (14) involves the undefined quantities $v_{i,j-1}$ and $q_{i,j-1}$. However, on one hand, discretisation of the natural boundary condition at pixel $(i, j-1)$ by forward differences provides the following equation:

$$v_{i,j} - v_{i,j-1} = q_{i,j-1} \tag{15}$$

On the other hand the natural boundary condition can be also discretised at pixel (i, j) by backward differences, leading to

$$v_{i,j} - v_{i,j-1} = q_{i,j} \tag{16}$$

Taking the mean of these forward and backward discretisations, we obtain:

$$v_{i,j} - v_{i,j-1} = \frac{q_{i,j-1} + q_{i,j}}{2} \tag{17}$$

Now, plugging Eq. (17) into Eq. (14), the undefined quantities actually vanish, and one obtains:

$$\begin{aligned} &-3v_{i,j} + (v_{i+1,j} + v_{i-1,j} + v_{i,j+1}) \\ =&\frac{p_{i+1,j} - p_{i-1,j} + q_{i,j+1} + q_{i,j}}{2} \end{aligned} \tag{18}$$

In other words, the stencil for the Laplacian is replaced by

$$\Delta v(x_i, y_j) \approx \begin{array}{|c|c|c|}\hline & 1 & \\ \hline 1 & -3 & 1 \\ \hline & & \\ \hline\end{array} \cdot\, v_{i,j}$$

and that for the divergence by

$$\mathrm{div}(p_{i,j}, q_{i,j}) \approx \frac{1}{2}\begin{array}{|c|c|c|}\hline \text{-1} & 0 & 1 \\ \hline\end{array} \cdot p_{i,j} + \frac{1}{2}\begin{array}{|c|}\hline 1 \\ \hline 1 \\ \hline 0 \\ \hline\end{array} \cdot q_{i,j} \tag{19}$$

The corresponding stencils for upper, left, and right borders are obtained by straightforward adaptations of this procedure.

When two neighbours are missing. Boundary pixels having exactly two neighbours outside Ω are either "corners" (e.g., $(i, j-1)$ and $(i+1, j)$ inside Ω, but $(i-1, j)$ and $(i, j+1)$ outside Ω) or "lines" (e.g., $(i-1, j)$ and $(i+1, j)$ inside Ω, but $(i, j-1)$ and $(i, j+1)$ outside Ω). For "lines", the natural boundary condition must be discretised four times (both forward and backward, on the two locations of missing data). Applying a similar rationale as in the previous case, we obtain the following stencils for "vertical" lines:

$$\Delta v(x_i, y_j) \approx \begin{array}{|c|c|c|}\hline & 1 & \\ \hline & -2 & \\ \hline & 1 & \\ \hline\end{array} \cdot v_{i,j}$$

and

$$\operatorname{div}(p_{i,j}, q_{i,j}) \approx \frac{1}{2}\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline\end{array} \cdot p_{i,j} + \frac{1}{2}\begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline -1 \\ \hline\end{array} \cdot q_{i,j} \quad (20)$$

A straightforward adaptation provides the stencils for the "horizontal" lines. Applying the same procedure for corners, we obtain, for instance for the "top-left" corner:

$$\Delta v(x_i, y_j) \approx \begin{array}{|c|c|c|}\hline & & \\ \hline & -2 & 1 \\ \hline & 1 & \\ \hline\end{array} \cdot v_{i,j}$$

and

$$\operatorname{div}(p_{i,j}, q_{i,j}) \approx \frac{1}{2}\begin{array}{|c|c|c|}\hline 0 & 1 & 1 \\ \hline\end{array} \cdot p_{i,j} + \frac{1}{2}\begin{array}{|c|}\hline 0 \\ \hline -1 \\ \hline -1 \\ \hline\end{array} \cdot q_{i,j} \quad (21)$$

Again, it is straightforward to find the other three discretisations for the other corner types.

When three neighbours are missing. In this last case, we discretise the boundary condition six times (forward and backward, for each missing neighbour). Most quantities actually vanish. For instance, for the case where only the right neighbour $(i+1, j)$ is inside Ω, we obtain the following stencils:

$$\Delta v(x_i, y_j) \approx \begin{array}{|c|c|c|}\hline & & \\ \hline & -1 & 1 \\ \hline & & \\ \hline\end{array} \cdot v_{i,j}$$

and

$$\operatorname{div}(p_{i,j}, q_{i,j}) \approx \frac{1}{2}\begin{array}{|c|c|c|}\hline 0 & 1 & 1 \\ \hline\end{array} \cdot p_{i,j} + \frac{1}{2}\begin{array}{|c|}\hline 0 \\ \hline 0 \\ \hline 0 \\ \hline\end{array} \cdot q_{i,j} \quad (22)$$

In the end, we obtain explicit stencils for all fourteen types of boundary pixels. Let us emphasise that, apart from 4-connectivity, we make no assumption about the shape of Ω.

Summarising the discretisation. The discretisation procedure defines a sparse linear system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Incorporating Neumann boundary conditions, the matrix $\boldsymbol{A}$ is symmetric, positive semidefinite, diagonal dominant and its null space contains the vector $e := [1, \ldots, 1]^{\mathrm{T}}$. In other words, $\boldsymbol{A}$ is a rank-1 deficient, singular matrix.

2.3 Iterative Krylov subspace methods

As indicated, in consequence of enormous memory costs, application of a direct solver to deal with the above linear system appears to be impractical for large images. Therefore, we propose an iterative solver to handle this problem.

Krylov subspace solvers are a modern class of iterative solvers designed for use with large sparse linear systems; for a detailed exposition see, e.g., Refs. [31, 41]. The main idea behind the Krylov approach is to search for an approximate solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is a large regular sparse matrix and $\boldsymbol{b} \in \mathbb{R}^n$, in a suitable low-dimensional (affine) subspace of $\mathbb{R}^n$ that is constructed iteratively.

This construction is in general not directly visible in the formulation of a Krylov subspace method, as these are often described in terms of a reformulation where $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is solved as an optimisation task. An important example is given by the classic conjugate gradient (CG) method of Hestenes and Stiefel [42] which is still an adequate iterative solver for problems involving sparse symmetric matrices of the kind in Eq. (14)①.

Conjugate gradient method. As it is of special importance for this work, let us briefly recall some properties of the CG method; a more technical, complete exposition can be found in many textbooks on numerical computing (see, e.g., Refs. [31, 41, 43, 44]).

Note that a useful implementation of CG is given in MATLAB. However, some knowledge of the technique is useful in order to understand some of its properties. Moreover, it is crucial for effective application of the CG method to be aware of its critical parameters. We now aim to make clear the relevant points.

① While in general also positive definiteness is required, this point is more delicate. We comment later on the applicability in our case.

The CG method requires a symmetric and positive definite matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. In its construction it combines the gradient descent method with the method of conjugate directions. It can be derived from making use of the fact that, for such a matrix, the solution of $\boldsymbol{Ax} = \boldsymbol{b}$ is exactly the minimum of the function:

$$F(\boldsymbol{x}) = \frac{1}{2}\langle \boldsymbol{x}, \boldsymbol{Ax}\rangle_2 - \langle \boldsymbol{b}, \boldsymbol{x}\rangle_2 \tag{23}$$

since

$$\nabla F(\boldsymbol{x}) = \boldsymbol{0} \quad \Leftrightarrow \quad \boldsymbol{Ax} = \boldsymbol{b} \tag{24}$$

here, $\langle\cdot,\cdot\rangle_2$ means the Euclidean scalar product.

Let us now denote the kth Krylov subspace by $\mathcal{K}_k$. Then, $\mathcal{K}_k := \mathcal{K}_k(\boldsymbol{A}, \boldsymbol{r}_0)$ is a subspace of $\mathbb{R}^n$ defined by

$$\mathcal{K}_k := \text{span}\left(\boldsymbol{r}_0, \boldsymbol{A}\boldsymbol{r}_0, \boldsymbol{A}^2\boldsymbol{r}_0, \ldots, \boldsymbol{A}^{k-1}\boldsymbol{r}_0\right) \tag{25}$$

This means $\mathcal{K}_k$ is generated from an initial residual vector $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{Ax}_0$ by successive multiplications by the system matrix $\boldsymbol{A}$.

Let us briefly highlight some important theoretical considerations. The nature of an iterative Krylov subspace method is that the computed approximate solution $\boldsymbol{x}_k$ is in $\boldsymbol{x}_0 + \mathcal{K}_k(A, \boldsymbol{r}_0)$, i.e., it is determined by the kth Krylov subspace. Here, the index k is also the kth iteration of the iterative scheme.

For the CG method, one can show that the approximate solutions $\boldsymbol{x}_k$ are optimal in the sense that they minimise the so-called energy norm of the error vector. Thus, if $\boldsymbol{x}^*$ is a solution of the system $\boldsymbol{Ax} = \boldsymbol{b}$, that $\boldsymbol{x}_k$ minimises $||\boldsymbol{x}^* - \boldsymbol{x}_k||_{\boldsymbol{A}}$ for the $\boldsymbol{A}$-norm $||\boldsymbol{y}||_{\boldsymbol{A}} := \sqrt{\boldsymbol{y}^{\mathrm{T}}\boldsymbol{Ay}}$. Note again that $\boldsymbol{x}_k$ must lie in the kth Krylov subspace. In other words, the CG method gives in the kth iteration the best solution available in the generated subspace. Since the dimension of the Krylov subspace increases in each step of the iteration, theoretical convergence is achieved at the latest after the nth step of the method if the solution is in $\mathbb{R}^n$.

Practical issues. A useful observation on Krylov subspace methods is that they can obviously benefit from a good educated guess of the solution for use as the initial iterate $\boldsymbol{x}_0$. Therefore, we consider $\boldsymbol{x}_0$ as an important open parameter of the method that should be addressed in a suitable way.

Moreover, an iterative method also requires the user to set a tolerance defining the *stopping criterion*: if the norm of the relative residual is below the tolerance, the algorithm stops.

However, there is *a priori* no means to say in which regime the tolerance has to be chosen. This is one of the issues that make reliable and efficient application of the method less than straightforward. It is one of the aims of our experiments to determine a reasonable tolerance for our application.

While our presentation of the CG method relates to ideal theoretical properties, in practice, numerical rounding errors appear and very large systems may suffer from severe convergence problems. Thus, *preconditioning* is recommended to ensure all beneficial properties of the algorithm, along with fast convergence. However, as it turns out, it requires a thorough study to identify the most useful parameters in the preconditioning method.

Let us note that the CG method is applicable even though our matrix $\boldsymbol{A}$ is just positive *semidefinite*. The positive definiteness is useful for avoiding division by zero within the CG algorithm. If $\boldsymbol{A}$ is positive semidefinite, theoretically it may happen that one needs to restart the scheme using a different initialisation. In practice this situation rarely occurs.

Preconditioning. The basic idea of preconditioning is to multiply the original system $\boldsymbol{Ax} = \boldsymbol{b}$ on the left by a matrix $\boldsymbol{P}$ such that $\boldsymbol{P}$ approximates $\boldsymbol{A}^{-1}$. The modified system $\boldsymbol{PAx} = \boldsymbol{Pb}$ is in general better conditioned and much more efficient to solve. For sparse $\boldsymbol{A}$, typical preconditioners are defined over the same sparse structure of entries of $\boldsymbol{A}$.

When dealing with symmetric matrices as in our case, incomplete Cholesky (IC) decomposition [45] is often used to construct a common and very efficient preconditioner for the CG method [46–48]. As a consequence of Ref. [29] we study here the application of the IC preconditioner and its modified version MIC.

Let us briefly describe the underlying ideas. The complete decomposition of $\boldsymbol{A}$ is given by $\boldsymbol{A} = \boldsymbol{LL}^T + \boldsymbol{F}$. If the lower triangular matrix $\boldsymbol{L}$ is allowed to have non-zero entries anywhere in the lower matrix, then $\boldsymbol{F}$ is the zero matrix and the decomposition is the standard Cholesky decomposition. However, in the context of sparse systems only the structure of entries in $\boldsymbol{A}$ is used in defining $\boldsymbol{L}$, so that the factorisation will be incomplete. Thus, in our case the lower triangular matrix $\boldsymbol{L}$ keeps the same non-zero pattern as that of the lower triangular part of $\boldsymbol{A}$. The general form of the preconditioning then amounts to the transformation from $\boldsymbol{Ax} = \boldsymbol{b}$ to $\boldsymbol{A}^p\boldsymbol{x}^p = \boldsymbol{b}^p$ with

$$\boldsymbol{A}^p = \boldsymbol{L}^{-1}\boldsymbol{AL}^{-T},\ \boldsymbol{x}^p = \boldsymbol{L}^{-T}\boldsymbol{x}, \text{and } \boldsymbol{b}^p = \boldsymbol{L}^{-1}\boldsymbol{b} \tag{26}$$

Practical issues. Let us identify another important computational parameter. The approach mentioned of taking the existing pattern in $\boldsymbol{A}$ as the sparsity pattern of $\boldsymbol{L}$ is often called IC(0). If one extends the sparsity pattern of $\boldsymbol{L}$ by additional non-zero elements (usually in the vicinity of existing entries) then the closeness between the product $\boldsymbol{L}\boldsymbol{L}^T$ and $\boldsymbol{A}$ may be potentially improved. This procedure is often called a numerical fill-in strategy IC(τ), with *drop tolerance*, where the parameter $\tau > 0$ describes a dropping criterion [31]. The approach can be described as follows: new fill-ins are accepted only if the elements are greater than a local drop tolerance τ. It turns out that the corresponding PCG method is applicable for positive semidefinite matrices [49, 50].

When dealing with a discretised elliptic PDE as in Eqs. (7) or (8), the *modified IC* (*MIC*) factorisation can lead to an even better preconditioner. For an overview of MIC see Refs. [43, 48]. The idea behind the modification is to force the preconditioner to have the same row sums as the original matrix $\boldsymbol{A}$. This can be accomplished by adding dropped fill-ins to the diagonal. The latter is known as MIC(0) and can be combined with the drop tolerance strategy to MIC(τ). We note that MIC can lead to possible pivot breakdowns. This problem can be circumvented by a global diagonal shift applied to $\boldsymbol{A}$ prior to determining the incomplete factorisation [51]. Therefore, the factorisation[①] of $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \alpha \operatorname{diag}(\boldsymbol{A})$ is performed, where $\alpha > 0$ and $\operatorname{diag}(\boldsymbol{A})$ is the diagonal part of $\boldsymbol{A}$. Note that the diagonal part of $\boldsymbol{A}$ never contains a zero value.

Adding fill-ins may obviously lead to a better preconditioner and a potentially better convergence rate. On the other hand, it becomes computationally more expensive to compute the preconditioner itself. Thus, there is a trade-off between speed and the improved convergence rate, an important issue upon which we will elaborate for our application.

2.4 On the FM-PCG normal integrator

Due to its local nature, the reconstructions computed by FM often have a lower quality compared to results of global approaches. On the other hand, the empowering effect of preconditioning the Poisson integration may still not suffice to achieve a high efficiency. The basic idea we follow now is that if one starts the PCG with a proper initialisation $\boldsymbol{x}_0$ obtained by FM integration, instead of the standard case $\boldsymbol{x}_0 = 0$, the PCG normal integrator could benefit from a significant speed-up. This idea, together with dedicated numerical evaluation using a well-engineered choice of computational parameters for the numerical PCG solver, is the core of our proposed method.

In the following, we first given the important building blocks and parameters of our algorithm, in Section 3. It will become evident how the *individual* methods perform and how they compare.

After that we will show in Section 4 that our proposed FM-PCG normal integrator in which *suitable building blocks are put together* is highly competitive, and in many instances superior to the state-of-the-art methods for surface normal integration.

3 Numerical evaluation

We now demonstrate relevant properties of several state-of-the-art methods for surface normal integration. For this purpose, we give a careful evaluation regarding the accuracy of the reconstruction, the influence of boundary conditions, flexibility to handle non-rectangular domains, robustness to noisy data, and computational efficiency—the main challenges for an advanced surface normal integrator. On the technical side, we note that the experiments were conducted on a i7 processor at 2.9 GHz.

Test datasets. To evaluate the proposed surface normal integrators, we provide examples of applications in gradient-domain image reconstruction (PET imaging, Poisson image editing) and surface-from-gradient (photometric stereo). Gradients of the "Phantom" and "Lena" images were constructed using finite differences, while both the surface and the gradient of the "Peaks", "Sombrero", and "Vase" datasets are analytically known, preventing any bias due to finite difference approximations.

We note that our test datasets demonstrate fundamental issues that one may typically find in gradient fields obtained from real-world problems: sharp gradients ("Phantom"), rapidly fluctuating gradients oriented in all grid directions in textured

① We denote the combined methods of MIC(τ) and the shifted incomplete Cholesky version as MIC(τ, α).

areas ("Lena"), and smoothly varying gradient fields ("Peaks"). The gradient field of the "Vase" dataset has a non-trivial computational domain.

3.1 Existing integration methods

Fast *and* accurate surface normal integrators are not abundant. For a meaningful assessment we compare our novel FM-PCG approach with the fast Fourier transform (FFT) method of Frankot and Chellappa [17] and the discrete cosine transform (DCT) extended by Simchony et al. [18] which are two of the most popular methods in use. Furthermore, we include the recent method of Harker and O'Leary [28] which relies on the formulation of the integration problem as a Sylvester equation. It is helpful to consider in a first step the building blocks of our approach, i.e., FM and CG-based Poisson integration separately. Hence, we also include in our comparison the FM method from Ref. [15]. As for Poisson integration, only Jacobi [9, 32] and Gauss–Seidel [27] iterations have been employed so far, so we consider Ref. [42] as a reference for CG-Poisson integration.

To highlight the differences between the methods, we start by comparing their algorithmic complexity, the type of admissible boundary conditions they admit, and the permissible the computational domain Ω: see Table 1. Algorithmic complexity is an indicator for the speed of a solver, while the admissible boundary conditions and the handling of non-rectangular domains influence its accuracy. The ability to handle non-rectangular domains improves also its computational efficiency.

The findings in Table 1 already indicate the potential usefulness of a mixture of FM and CG-Poisson approaches as both are free of constraints in the last two criteria and so their combination may lead to a reasonably computationally efficient Poisson solver. Although other methods have their strengths in algorithmic complexity and in the application of boundary conditions (apart from FFT), we see that the flexible handling of domains is a fundamental task and a key requirement of an ideal solver for surface normal integration.

Table 1 Comparison of five existing fast and accurate surface normal integration methods based on three criteria: their algorithmic complexity w.r.t. the number n of pixels inside the computational domain Ω (the lower the better), the type of boundary condition (BC) they use (free boundaries are expected to reduce bias), and the permitted shape of Ω (handling non-rectangular domains can be important for accuracy and algorithmic speed)

Method	Ref.	Complexity	BC	Non-rect.
FFT	[17]	$n \log n$	Periodic	No
DCT	[18]	$n \log n$	**Free**	No
FM	[15]	$n \log n$	**Free**	**Yes**
Sylvester	[28]	$n^{\frac{3}{2}}$ ①	**Free**	No
CG-Poisson	[42]	n^3 ②	**Free**	**Yes**

①Assuming Ω is square. For rectangular domains of size $n_r \times n_c$, the complexity is $O(n_c^3)$.

②Without using preconditioning techniques.

3.2 Stopping criterion for CG-Poisson

Amongst the considered methods, the Poisson solver (conjugate gradient method), where solving the discrete Poisson equation Eq. (7) corresponds to a linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, is the only iterative scheme. As indicated in Section 2.3, a practical solution can be reached quickly after a small number k of iterations, but k cannot be predicted exactly. The general stopping criterion for an iterative method can be based on the *relative residual* $(\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|)/\|\boldsymbol{b}\|$ which we analyse in this paragraph.

To guarantee the efficiency of the CG-Poisson solver it is necessary to define the number k of iterations depending on the quality of the reconstruction in the iterative process. To tackle this issue we compared the MSE③ and the relative residual during each CG iteration. As the solution of the linear system Eq. (14) is not unique, an additive ambiguity $v \mapsto v + c, c \in \mathbb{R}$ in the integration problem (c is the "integration constant") occurs. Therefore, in each numerical experiment we chose the additive constant c which minimises the MSE, for fair comparison. To determine a proper relative residual, we considered the datasets "Lena", "Peaks", "Phantom", "Sombrero", and "Vase" on rectangular and non-rectangular domains. All test cases showed results similar to the graphs in Fig. 1 for the reconstruction of the "Sombrero" surface (see Fig. 2).

In this experiment the iterative solver CG-Poisson was stopped when the relative residual was lower than 10^{-6}. However, it can be seen clearly that after around iteration 250, the quality measured by the MSE cannot be improved and therefore using more than 400 iterations is redundant. This numerical steady state of the MSE and therefore of the residual occurs when the relative residual is between 10^{-3}

③The mean squared error (MSE) is used to quantify the error of the reconstruction. We employed it to estimate the amount of the error contained in the reconstruction compared to the original.

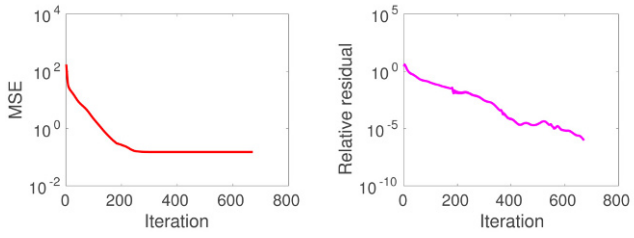


Fig. 1 MSE vs. relative residual during CG iterations, for the "Sombrero" dataset. Although arbitrary relative accuracy can be reached, it is not useful to go beyond a 10^{-3} residual, since such refinements have very small impact on the quality of the reconstruction, as shown by the MSE graph. Similar results were obtained for all datasets used in this paper. Hence, we set as stopping criterion a 10^{-4} relative residual, which can be considered as "safe".

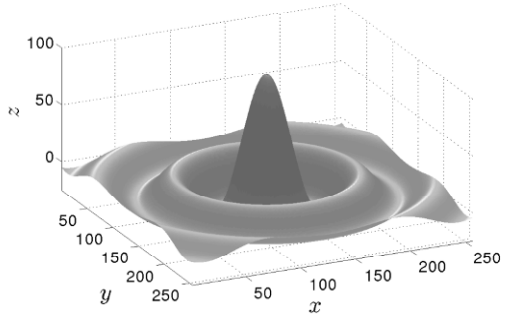


Fig. 2 "Sombrero" surface (256×256) used in this experiment, whose gradient can be calculated analytically. Note that the depth values are periodic on the boundaries.

and 10^{-4}, so we use the suitable and "safe" stopping criterion of 10^{-4} in subsequent experiments.

3.3 Accuracy of the solvers

First we analyse the general quality of the methods listed in Table 1 for the "Sombrero" dataset over a domain of size 256×256. In this example the gradient can be calculated analytically and furthermore the boundary conditions are periodic. Table 1 makes it obvious that all methods have no restrictions and consequently no discrimination.

Basically all methods provide a satisfactory reconstruction, and only FM produces a less accurate solution: see Figs. 3 and 4. This can be seen more easily in Table 2, where the values of the measurements① of MSE and SSIM and the CPU time (in second) are given. The accuracy of all methods is similar, although the solution for FM, with 1.16 for MSE and 0.98 for SSIM, is slightly worse.In contrast the CPU time varies strongly, and

①We tested the two common measurements MSE and SSIM. A superior reconstruction has value closer to zero for MSE and a value closer to one for SSIM. The structural similarity (SSIM) index is a method for predicting the perceived quality of an image, see Ref. [52].

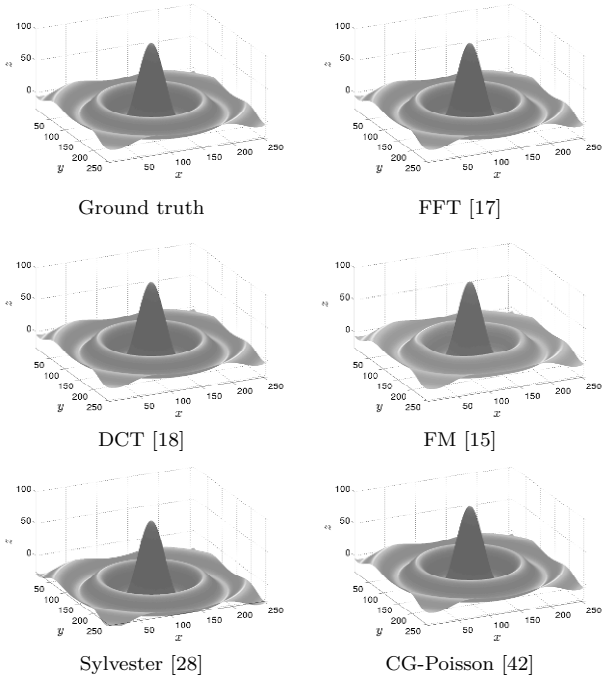


Fig. 3 Results on the "Sombrero" dataset (cf. Table 2).

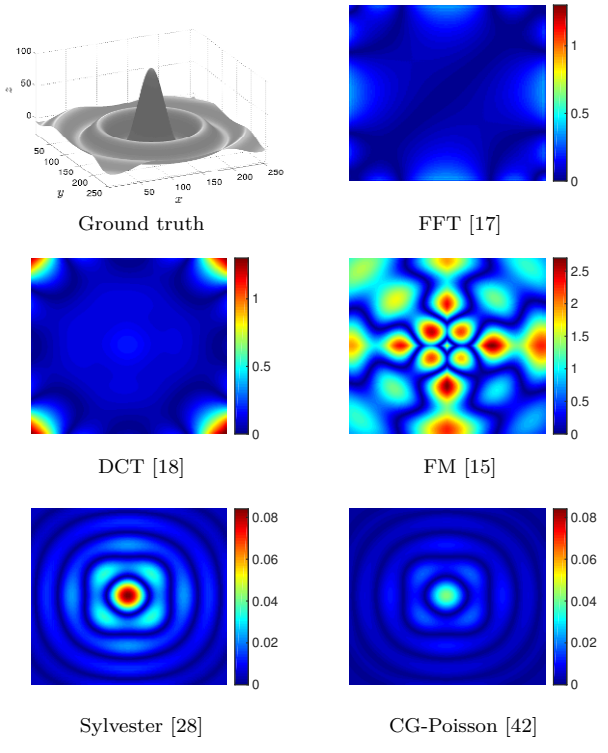


Fig. 4 Absolute errors for the "Sombrero" dataset between the ground truth and the numerical result of each method; see Table 2. Note the different scales of the plots. To highlight the differences between all methods we use three different scales. The absolute error map of FM has its own scale due to the fact that the MSE and the maximum error differ compared to the other methods. For FFT and DCT, Sylvester and CG respectively, which have similar values for the MSE and the maximum error, we used the same scale to point out the differences.

Table 2 Results on the "Sombrero" dataset (256 × 256). As expected, all methods provide reasonably accurate solutions. However, the FM result is slightly less accurate: this is due to error accumulation by the *local* nature of FM, while the other methods are *global*. The reconstructed surfaces are shown in Fig. 3

Method	MSE (px)	SSIM	CPU (s)
FFT [17]	0.01	**1.00**	$<$ **0.01**
DCT [18]	0.04	**1.00**	0.01
FM [15]	1.00	0.98	0.07
Sylvester [28]	1.8×10^{-4}	**1.00**	0.18
CG-Poisson [42]	$\mathbf{4.6 \times 10^{-5}}$	**1.00**	1.06

in this case FFT and DCT, which need around 0.01 s, are unbeatable. The time for FM and Sylvester is in a reasonable range, and only the standard CG-Poisson taking around 1.06 s being too slow and inefficient. For problems of surface reconstruction under these conditions, the choice of a solver is fairly easy: the frequency domain methods FFT and DCT are best.

3.4 Influence of boundary conditions

The handling of boundary conditions is a necessary issue which cannot be ignored. As we will show, different boundary conditions lead to surface reconstructions of different accuracy. The assumption of Dirichlet, periodic or homogeneous Neumann boundary conditions is often not justified and may even be unrealistic in some applications. A better choice is to use "natural" boundary conditions [2] of Neumann type.

The behaviour of the discussed solvers for unjustified boundary conditions, particularly for FFT, is illustrated by the "Peaks" dataset in Figs. 5 and 6 and the associated Table 3. Almost all methods provide good reconstructions; the FM result is also acceptable. Only FFT, with 7.19 for MSE and 0.96 for SSIM, is strongly inferior and unusable for this real surface, as its accuracy is too low.

Our results show that FFT-based methods enforcing periodic boundary conditions can be discarded from the list of candidates for an ideal

Table 3 Results on the "Peaks" dataset (128 × 128). Methods enforcing periodic BC fail to provide a good reconstruction. The reconstructed surfaces are shown in Fig. 5

Method	MSE (px)	SSIM	CPU (s)
FFT [17]	7.19	0.96	$<$ **0.01**
DCT [18]	0.09	**1.00**	$<$ **0.01**
FM [15]	0.80	0.99	0.03
Sylvester [28]	**0.02**	**1.00**	0.05
CG-Poisson [42]	**0.02**	**1.00**	0.29

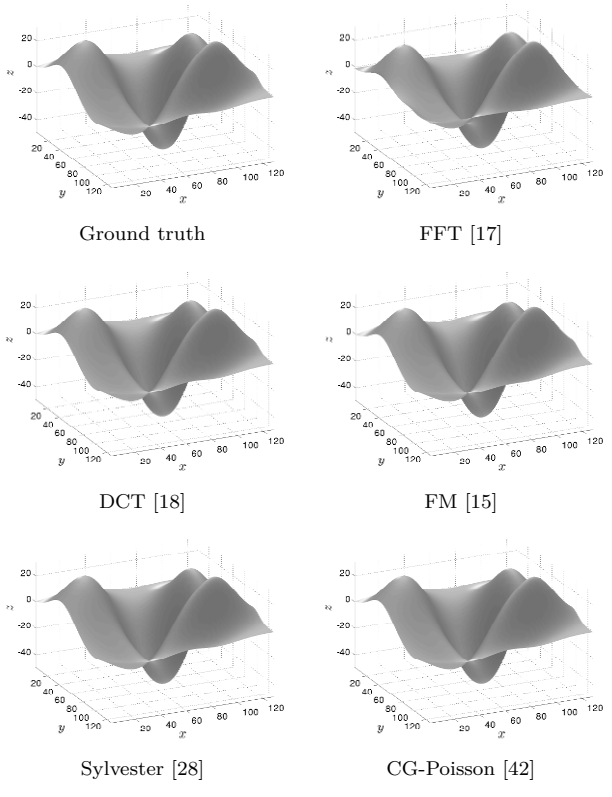


Fig. 5 Results on the "Peaks" dataset (see Table 3).

solver. Once again CG-Poisson is a very accurate integrator, but DCT and Sylvester are much faster and provide useful results. However, we may point out again in advance that enforcing the domain Ω to be rectangular may lead to difficulties w.r.t. the transition from foreground to background of an object.

3.5 Influence of noisy data

A key point in many applications is the question of the influence of noise on the quality of the reconstructions provided by different methods. Usually, the correctness of the given data, without noise, cannot be guaranteed. Therefore, it is essential to have a robust surface normal integrator with respect to noisy data.

To study the influence of noise, we should consider a dataset which, apart from noise, is perfect. Based on this aspect, a very reasonable test example is the "Sombrero" dataset; see Fig. 2. The advantage of "Sombrero" is that the gradient of this object is known analytically, not just approximately. Furthermore, the computational domain Ω is rectangular and the boundary conditions are periodic. For this test we added Gaussian noise with a standard deviation σ varying from 0% to 20%

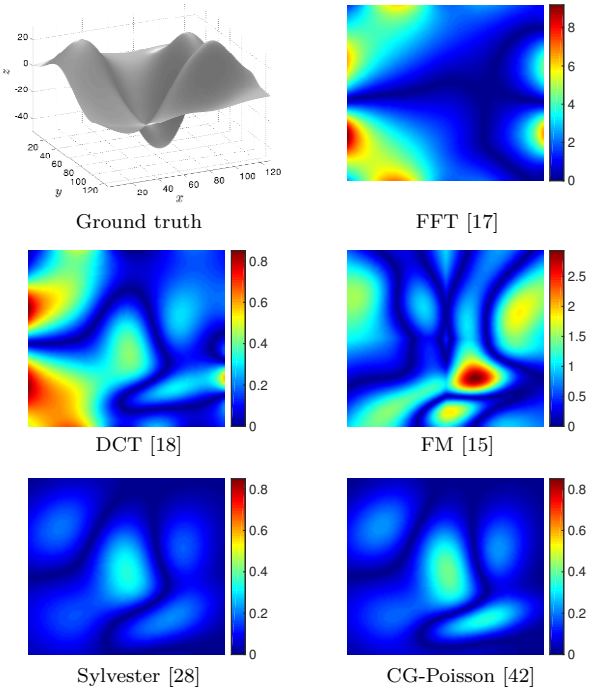


Fig. 6 Absolute errors for the "Peaks" dataset between the ground truth and the numerical result of each method; see Table 3. Note the different scales of the plots. To highlight the differences between all methods we consider three different scales. The absolute error maps of FFT and FM have their own scales due to the fact that the MSE and the maximum error are very different in contrast to the other methods. For DCT, Sylvester, and CG, which have similar values for the MSE and the maximum error, we used the same scale to point out the differences.

of $\|[p,q]^{\mathrm{T}}\|_\infty$ to the known gradients①.

The graph in Fig. 7, which compares the MSE versus the standard deviation of Gaussian noise, indicates the robustness of the tested methods. The best performance is achieved by FFT, DCT, and CG-Poisson, even for strong noisy data with a standard deviation of 20%.

The results of Sylvester are similar, but the method suffers from weaknesses in examples with noise of higher standard deviation, i.e., larger than 10%.

As the FM integrator accumulates errors during front propagation, we observe, as expected, for highly noisy data this integrator is no longer a useful choice.

To conclude, if the accuracy of the given data is not known then FFT, DCT, and CG-Poisson are the safest integrators.

①In the context of photometric stereo, it would be more realistic to add noise to the input images rather than to the gradients [53]. Nevertheless, evaluating the robustness of integrators given noisy gradients remains useful in order to compare their intrinsic properties.

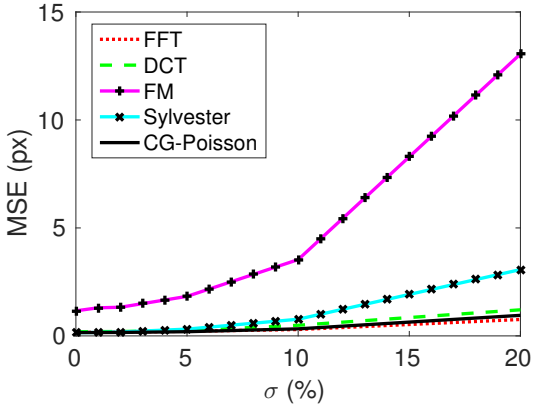


Fig. 7 MSE as a function of the standard deviation of a Gaussian noise, expressed as percentage of the maximal amplitude of the gradient, added to the gradient. FFT, DCT, and CG-Poisson methods provide the best results for different levels of Gaussian noise. The Sylvester method leads to reasonable results for a noise level lower than 10%. Since the FM approach propagates information in a single pass, it obviously also propagates errors, resulting in reduced robustness compared to all other approaches.

3.6 Handling non-rectangular domains

In this experiment we consider the situation when the gradient values are only known on a non-rectangular part of the grid. Applying methods needing rectangular grids [17, 18, 28] requires empirically fixing the values $[p,q] := [0,0]$ outside Ω (see Fig. 8), inducing a bias.

This can be explained as follows: filling the gradient with null values outside Ω creates discontinuities between the foreground and background, preventing one from obtaining reasonable results, since all solvers considered here are intended to reconstruct smooth surfaces. This problem is illustrated in Figs. 9 and 10 for the "Vase" dataset; Table 4 gives corresponding values for MSE and SSIM.

The methods can be classified into two groups. In contrast to FM and CG-Poisson, FFT, DCT, and Sylvester methods, which cannot handle flexible

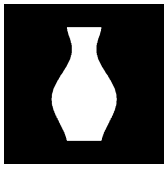


Fig. 8 Mask for the "Vase" dataset. The gradient values are only known on a non-rectangular part Ω of the grid, represented by the white region. The FM and the CG-Poisson integrators can handle easily any form of domain Ω. In contrast FFT, DCT, and Sylvester rely on a rectangular domain and therefore the values $[p,q] := [0,0]$ need to be fixed outside Ω, in the black region.

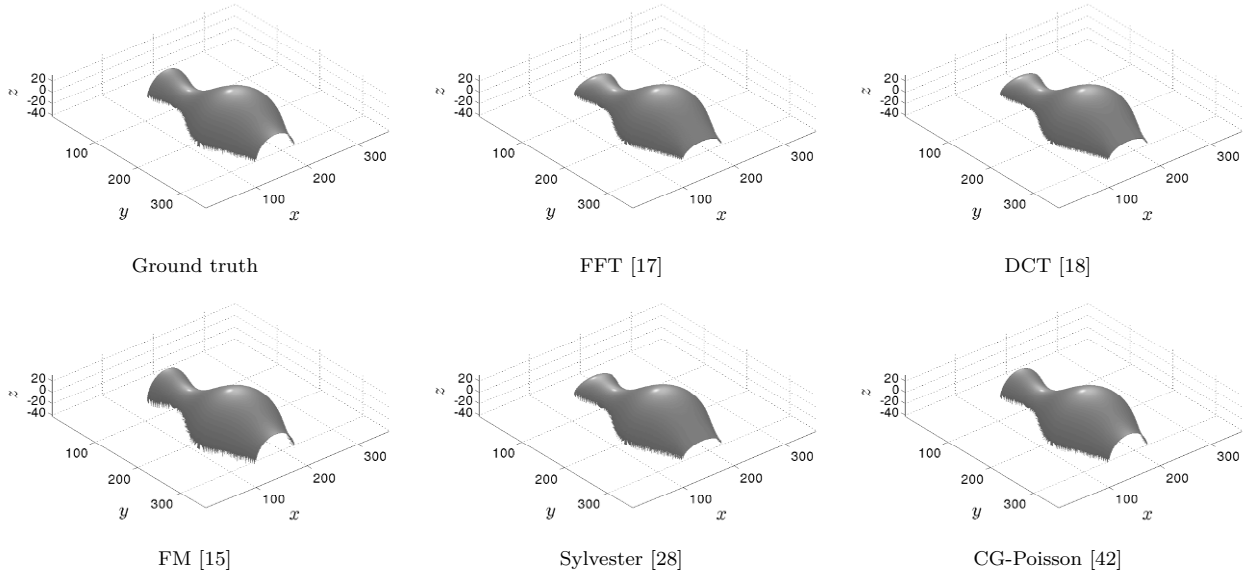


Fig. 9 Results on the "Vase" dataset; see Table 4.

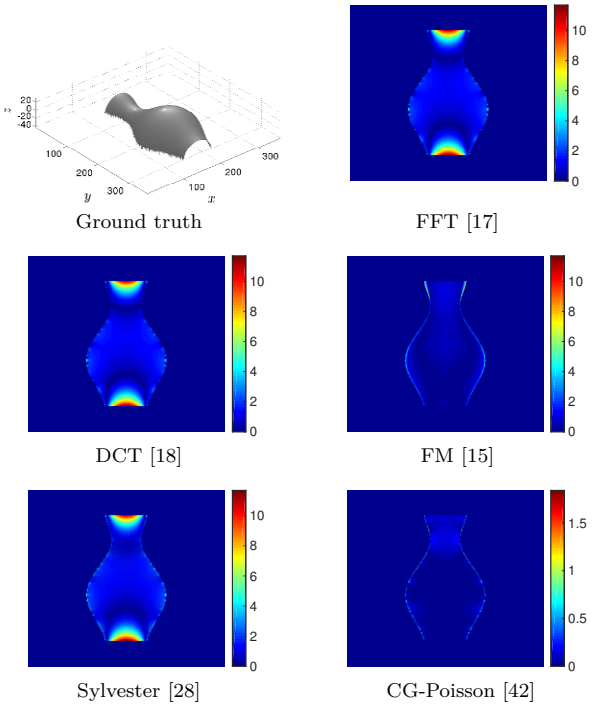


Fig. 10 Absolute errors for the "Vase" dataset between the ground truth and the numerical result of each method; see Table 4. Note the different scales of the plots. To highlight the differences between all methods we use two different scales. The absolute error maps of FFT, DCT, Sylvester, and FM have the same scale due as their maximum errors are very close. For CG we use a different scale to point out the differences.

domains, provide inaccurate reconstructions which are not useful. The non-applicability of these methods is a considerable problem, since real-world input images for 3D reconstruction are typically located within a photographed scene. This requires the flexibility to tackle non-rectangular domains, while providing accurate (and efficient) reconstruction.

Table 4 Results on the "Vase" dataset (320×320). Methods dedicated to rectangular domains are clearly biased if Ω is not rectangular. The corresponding reconstructed surfaces are shown in Fig. 9

Method	MSE (px)	SSIM	CPU (s)
FFT [17]	5.71	0.99	**0.01**
DCT [18]	5.69	0.99	0.02
FM [15]	0.71	**1.00**	0.15
Sylvester [28]	5.99	0.98	0.36
CG-Poisson [42]	**0.01**	**1.00**	0.52

This experiment shows that all methods except FM and CG-Poisson are not applicable as an ideal, high-quality normal integrator for many applications.

Let us note that we also have shown that FM and CG-Poisson have complementary properties and disadvantages: the former is fast but inaccurate, and the latter is slow but accurate. This clearly motivates the combination of FM as initialisation, and a Krylov-based Poisson solver: these should combine to give a fast and accurate solver.

3.7 Summary of the evaluation

In the previous experiments we tested different scenarios which arise in real-world applications. It was found that boundary conditions and noisy data may have a strong effect on 3D reconstruction. If rectangular domains can be considered, the DCT method seems to be a realistic choice of a normal integrator followed by Sylvester and CG-Poisson.

In fact the first is unbeatably fast. However, the importance of handling non-rectangular domains, which is a practical issue in many industrial applications, cannot be underestimated. This situation leads to inaccuracies in the reconstructions of DCT and Sylvester methods. In this context FM and CG-Poisson methods achieve better results.

One can observe a certain lack of robustness w.r.t. noise of the FM integrator, especially along directions not aligned with the grid structure: see also the results in Ref. [29]. This is 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.

Due to the possibly non-rectangular nature of the domains we aim to tackle, we cannot use fast Poisson solvers as, e.g., in Ref. [18] to solve the discrete Poisson equations numerically. Instead, we explicitly construct linear systems and solve them using the CG solver as often done by practitioners. Nevertheless, the unmodified CG-Poisson solver is still quite inefficient.

4 Accelerating CG-Poisson

Let us now demonstrate the advantages of the proposed FM-PCG approach compared to other state-of-the-art methods. In doing so, we give a careful evaluation of all the components of our novel algorithm.

4.1 Preconditioned CG-Poisson

In a first step we analyse the behaviour of the CG solver when applying an additional preconditioner intended to improve the condition number and convergence speed w.r.t. the number k of iterations, thus reducing time to reach the stopping criterion.

As examples of actual preconditioners, we examined① IC(τ) and MIC(τ, α) for the test dataset "Phantom" (see Fig. 11) for different input sizes. It was observed that MIC(τ, α^*) beats IC(τ) if we used $\alpha^* = 10^{-3}$ for the global diagonal shift②. In the following, for simplicity we write MIC(τ)* for MIC(τ, α^*)③. The results for MIC(0)*, without a fill-in strategy, are shown in Table 5 (third column) and demonstrate its usefulness compared to the non-preconditioned CG-Poisson (second column). By using MIC(0)* we save many iterations, greatly reducing the time taken: for example one can save around 2700 iterations and thus more than 1250 s for an image of size 4096×4096.

Fig. 11 "Phantom" image used in this experiment. Its gradient is unknown, so we approximate it numerically by first-order forward differences. We used this dataset for comparing preconditioners, for different image sizes, from 64×64 to 4096×4096.

Now we show how useful a fill-in strategy can be. In columns four to eight we tested different fill-in strategies from MIC(10^{-1})* to MIC(10^{-5})*. A closer examination of Table 5 shows that MIC(10^{-3})* provides the best balance between the time needed to compute the preconditioner and the time needed to apply PCG. As an example, again for the image of size 4096×4096, we can reduce the number of iterations from 247 to 80, taking around 170 s instead of 290 s.

Thus, the application of preconditioning, here shifted MIC, seems to be useful in accelerating the CG-Poisson integrator, but it is not sufficient to be competitive with common fast methods. However, we will see that with proper initialisation, this standard preconditioner can already be considered to be as efficient.

4.2 Appropriate initialisation

The suggested preconditioned CG-Poisson (PCG-Poisson) method is not widely known in computer vision, although this practical method is surely not new and commonly used in numerical computing. However, we propose a novel scheme for the surface normal integration (SNI) task, using an appropriate

① As pivot breakdowns for MIC(τ) are possible, we considered the shifted MIC version MIC(τ, α). All methods are predefined functions in MATLAB.

② This is an experimentally determined value.

③ If the value τ in MIC(τ)* tends to zero then the preconditioned matrix is more dense, with more non-zero elements. As a consequence, the preconditioner is better; however it costs more time to compute the preconditioner itself.

Table 5 Number of iterations and CPU time required to reach a 10^{-4} relative residual for the conjugate gradient algorithm, using the shifted modified incomplete Cholesky (MIC) preconditioner ($\alpha^* = 10^{-3}$) with different drop tolerances and different "Phantom" sizes. The 10^{-3} drop tolerance is the one which provides the fastest results. Using a larger drop tolerance allows a reduced number of required iterations, but the time used for computing the preconditioner dramatically increases. Note that we were unable to compute the preconditioner MIC(10^{-5})* for the 4096^2 dataset, because 32 GB of memory was insufficient

Size	No precond.		MIC(0)*		MIC(10^{-1})*		MIC(10^{-2})*		MIC(10^{-3})*		MIC(10^{-4})*		MIC(10^{-5})*	
	It.	CPU (s)	It.	CPU (s)	It.	CPU (s)	It.	CPU (s)	It.	CPU (s)	It.	CPU (s)	It.	CPU (s)
64^2	131	0.04	19	**0.01**	19	**0.01**	10	**0.01**	5	**0.01**	**4**	**0.01**	**4**	0.02
128^2	236	0.14	29	0.05	29	0.05	15	**0.04**	9	**0.04**	**6**	0.06	7	0.12
256^2	432	0.86	40	0.30	40	0.26	23	0.21	11	**0.18**	**7**	0.25	11	0.53
512^2	604	4.75	70	1.43	70	1.50	28	**0.88**	18	0.89	**13**	1.22	14	2.29
1024^2	1059	35.55	91	7.38	91	7.52	49	5.15	30	**5.02**	**19**	5.96	24	11.83
2048^2	1718	233.49	160	49.89	160	49.89	79	31.29	49	**28.76**	**34**	35.09	39	65.06
4096^2	2969	1577.81	247	290.93	247	290.54	134	196.1	80	**171.44**	**41**	173.5	N/A	N/A

initialisation to decrease the number of iterations and reduce the run time cost. Our proposed method consists of two steps: in a first step the FM solution is computed in a fast and efficient way; after that, the Krylov-based technique with shifted modified incomplete Cholesky (MIC) is applied.

To show the effect of the new FM initialisation, the test for the "Phantom" dataset was repeated and evaluated anew; see Table 6. Starting from the FM solution, which needs comparatively short computation time (see Table 7) even for large images, gives a dramatic speed-up.

A closer look at Tables 5 and 6 shows a significant difference, even without a fill-in strategy (compare both third columns). At first, let us consider the case without preconditioner: starting with the trivial solution leads to a constant increase in iterations (factor around 1.7) as the image size increases concomitantly. In contrast the number of iterations increases very slowly when using FM initialisation. The effect of this phenomenon is a notable, strong time cost reduction for large data: for 512×512 images, we can save more than 2 s (from 4.75 to 2.48 s), and for 4096×4096 images the time can be reduced from 1578 to 233 s.

Using additional preconditioning leads to similar results. Testing anew MIC(τ)* with MIC(10^{-1})* to MIC(10^{-5})* shows once more that MIC(10^{-3})* provides the best results; see Table 6. Using FM initialisation greatly reduces the required iterations to reach the stopping criterion and therefore the combination of FM and shifted MIC leads to fast reconstructions. In the case of an image of size 4096×4096, the novel approach, including the time taken to perform FM performing of 21.79 s (see Table 7), saves around 100 s (from 171 to 74 s) and 71 iterations compared with the trivial initialisation and MIC(10^{-3})*.

Finally, using the novel approach instead of the standard CG-Poisson solver leads to a significant speed-up; see Table 8. Without considering the computation of the FM initialisation, the construction of the system and the preconditioner, the time to purely solve the system is vastly reduced from 1552 to 19 s. The findings of this experiment show impressively that choosing FM as

Table 6 Number of iterations and CPU time for applying the PCG algorithm, starting from the FM solution rather than from the trivial state. The indicated CPU time includes the time for computing the FM initialisation. Using FM as an initial guess saves many computations: the time to solve the 4096^2 problem is reduced from 26 min (with neither FM initialisation nor preconditioning, see second column in Table 5, to 1 min (with FM initialisation and preconditioning, see column 6)

Size	No precond.		MIC(0)*		MIC(10^{-1})*		MIC(10^{-2})*		MIC(10^{-3})*		MIC(10^{-4})*		MIC(10^{-5})*	
	It.	CPU (s)	It.	CPU (s)	It.	CPU (s)	It.	CPU (s)	It.	CPU (s)	It.	CPU (s)	It.	CPU (s)
64^2	119	0.05	16	**0.02**	16	**0.02**	8	**0.02**	4	**0.02**	**3**	**0.02**	**3**	0.04
128^2	210	0.17	25	0.10	25	0.10	13	**0.09**	7	**0.09**	**5**	0.12	6	0.16
256^2	240	0.74	35	0.32	35	0.31	15	0.24	7	**0.23**	**5**	0.30	6	0.54
512^2	281	2.48	36	1.18	36	1.21	15	**0.92**	9	0.93	**5**	1.07	6	2.22
1024^2	316	12.80	40	5.19	40	5.20	18	4.06	9	**3.97**	**5**	4.92	8	9.28
2048^2	339	55.44	45	23.14	45	23.28	19	13.08	9	**17.12**	**5**	21.47	8	39.91
4096^2	349	232.95	46	98.70	46	99.30	19	76.34	9	**74.04**	**5**	107.24	N/A	N/A

Table 7 CPU time to perform FM on the "Phantom" dataset of different sizes

Size	64^2	128^2	256^2	512^2	1024^2	2048^2	4096^2
CPU (s)	< 0.01	0.03	0.08	0.28	1.20	5.14	21.79

Table 8 Division of CPU time between system construction, preconditioning and system resolution, for the 4096^2 example. Knowing that the system and the preconditioner can often be precomputed, this makes even more obvious the gain one can expect by choosing an appropriate initialisation such as the FM result. CG refers to the resolution of the system by conjugate gradient, and +CG to accelerated resolution by using FM initialisation (time does not include the 21.79 s required for FM)

	Syst. constr.	Precond.	CG	+CG
No precond.	25.85	0	1551.96	**185.31**
MIC(10^{-3})*	25.85	7.50	138.09	**18.90**

initialisation accelerates the method greatly when it comes to standard preconditioners like (shifted modified) incomplete Cholesky. Thus, we believe that our novel FM-PCG method with shifted MIC preconditioning is a relevant contribution to the field of fast and accurate surface normal integrators.

4.3 Evaluation of the FM-PCG solver

To clarify the strength of our proposed FM-PCG solver against the standard FFT and DCT solvers and the "Sylvester" method of Harker and O'Leary, we use MSE to evaluate the reconstructions of the datasets "Phantom", "Lena", "Peaks", and "Vase" on rectangular and non-rectangular domains. At first we examine the "Phantom", "Lena", and "Peaks" datasets on a rectangular domain in Tables 9–11. All examples contain the natural boundary equation; "Phantom" and "Lena" have sharp gradients and are more realistic.

It should be clear that FFT and DCT are the fastest methods, but the quality of FFT is inadequate and the results are unusable. Furthermore, it can be seen that the FM-PCG solver is the best integrator for sharp gradients (see Table 10).

Finally, the method with the best speed–quality balance on rectangular domains is probably DCT,

Table 9 Results on the "Phantom" dataset (1024×1024)

Method	MSE (px)	CPU (s)
FFT [17]	138.6	**0.06**
DCT [18]	**127.31**	0.13
FM [15]	163.13	1.20
Sylvester [28]	169.41	5.78
FM-PCG	127.89	4.23

Table 10 Results on the "Lena" dataset (512×512)

Method	MSE (px)	CPU (s)
FFT [17]	402.37	**0.02**
DCT [18]	132.08	0.03
FM [15]	509.15	0.28
Sylvester [28]	113.92	0.71
FM-PCG	**94.07**	1.24

Table 11 Results on the "Peaks" dataset (128×128)

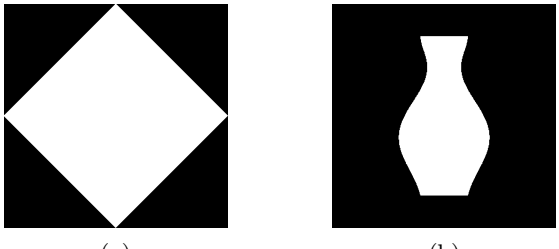
Method	MSE (px)	CPU (s)
FFT [17]	7.19	$<$ **0.01**
DCT [18]	0.09	$<$ **0.01**
FM [15]	0.8	0.03
Sylvester [28]	**0.01**	0.07
FM-PCG	0.02	0.07

followed by Sylvester and our proposed FM-PCG solver. However, as already mentioned, simple rectangular domains are quite unrealistic in many applications in science and industry. Hence, we analyse in Tables 12–15 the results on flexible domains, as shown in Fig. 12. The given CPU time includes FM initialisation.

All experiments show the expected behaviour of the employed methods. The FM-PCG solution has by far the best quality. It is even faster than the Sylvester method. An assessment in relation to the best balance of speed versus quality is not easy and depends on the exact application. If speed is of secondary importance then the best choice is FM-PCG, otherwise DCT.

4.4 Real-world photometric stereo data

The previous examples are rather simple. For this reason we consider a more realistic real-world application in photometric stereo, which definitely contains noisy data. We used the "Scholar"



(a) (b)

Fig. 12 Masks for the "Phantom", "Lena", "Peaks", and "Vase" datasets. It should be noted that FM and CG-Poisson work only on the Ω represented by the white regions. By contrast, FFT, DCT, and Sylvester work on the whole rectangular domain. (a) The synthetic mask used for "Phantom", "Lena", and "Peaks" datasets. (b) Realistic mask for the "Vase" dataset.

Table 12 Results on the "Phantom" dataset for the non-rectangular domain in Fig. 12(a)

Method	MSE (px)	CPU (s)
FFT [17]	351.72	**0.06**
DCT [18]	309.72	0.14
FM [15]	162.43	0.70
Sylvester [28]	348.98	5.54
FM-PCG	**131.02**	2.06

Table 13 Results on the "Lena" dataset for the non-rectangular domain in Fig. 12(a)

Method	MSE (px)	CPU (s)
FFT [17]	199.59	**0.01**
DCT [18]	149.00	0.03
FM [15]	444.02	0.19
Sylvester [28]	175.82	0.70
FM-PCG	**123.64**	0.65

Table 14 Results on the "Peaks" dataset for the non-rectangular domain in Fig. 12(a)

Method	MSE (px)	CPU (s)
FFT [17]	15.69	$<$ **0.01**
DCT [18]	7.23	$<$ **0.01**
FM [15]	0.86	0.01
Sylvester [28]	7.20	0.06
FM-PCG	**0.03**	0.03

Table 15 Results on the "Vase" dataset for the non-rectangular domain in Fig. 12(b)

Method	MSE (px)	CPU (s)
FFT [17]	5.71	**0.01**
DCT [18]	5.69	0.02
FM [15]	0.71	0.06
Sylvester [28]	5.99	0.38
FM-PCG	**0.03**	0.14

dataset①, which consists of 20 images of a Lambertian surface, taken from the same angle of view but under 20 known, non-coplanar lightings (see Fig. 13).

The normals and the albedo were calculated using the classical photometric stereo approach of Woodham [3]. Then, we integrated the normals using the different solvers. Eventually, we a posteriori recomputed the normals by finite differences from the recovered depth map, before "reprojecting" the images using the estimated shape and albedo. By comparing the initial images with the reprojected ones, we obtain two criteria (MSE and SSIM) for evaluating the methods on each image. The results shown in Table 16 are the mean of the 20 corresponding values.

Once again FM-PCG is the most accurate integrator and is as fast as the Sylvester method. Nevertheless, the fast computational time of DCT was unbeatable.

① http://vision.seas.harvard.edu/qsfs/Data.html

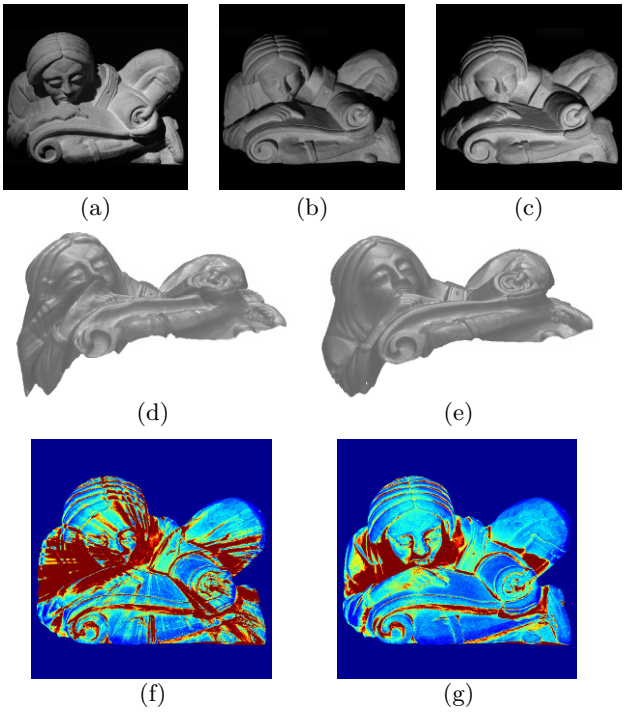


Fig. 13 Application to photometric stereo (PS). (a)–(c) Three images (among 20), of size 1070×1070, acquired from the same point of view but under different lightings. After estimating the surface normals by PS, we integrated them by (d) FM, before (e) refining this initial guess by PCG iterations. The full integration process required a few seconds. (f)–(g) MSE (in pixel) of the reprojected images, computed from the surface estimated by (f) FM and (g) FM-PCG. (blue is 0, and red is > 1000). Due to the local nature of FM, radial propagation of errors is visible. After correction by CG, such artefacts are eliminated. Remaining bias is due to shadows. These results are experimentally compared with existing methods in Table 16.

Table 16 Results on the PS dataset. Our method (initialisation by FM, then refinement by PCG from this initial guess) provides the most accurate results. We show the CPU time, as well as the mean MSE and SSIM for the 20 reprojected images

Method	MSE (px)	SSIM	CPU (s)
FFT [17]	365.43	0.86	**0.09**
DCT [18]	330.55	0.87	0.15
FM [15]	582.65	0.78	0.45
Sylvester [28]	377.68	0.74	5.81
FM-PCG	**286.69**	**0.88**	6.25

4.5 Handling outliers

Let us now consider the case of standard photometric stereo applied to surfaces whose reflectance incorporates an additive non-Lambertian

component (specularities). As can be seen from Fig. 14 and Table 17, all the integration methods we consider here are by their nature highly sensitive to outliers.

In order to handle such outliers, we replace the classic PI model in Eq. (7) by the modified model in Eq. (11). As already pointed out, all the methods relying on the Poisson equation can be adapted to this model. Therefore, we can employ here the FFT [17], DCT [18], and our new FM-PCG methods. We found that using these modified inputs for the other SNI methods, such as FM [15] and Sylvester [28], also yields improved results. Hence, our improved model can be considered as a *generic* improvement for use with existing SNI methods, enforcing robustness w.r.t. outliers. This is illustrated in Fig. 15 and Table 18.

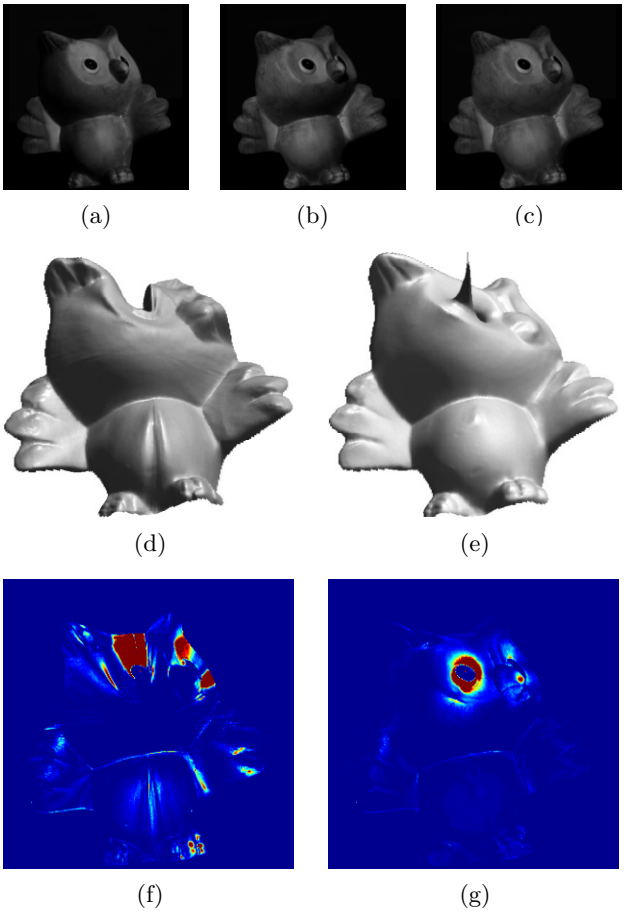


Fig. 14 (a)–(c) Three (out of 12) real-world images, of size 320 × 320, of a photometric stereo dataset. The eyes of the owl are highly specular. This induces a bias in the reconstructions, as shown in the reconstructions using (d) FM or (e) the proposed FM-PCG integrator. (f)–(g) The corresponding MSE of the reprojected images shows that the bias is very localized (blue is 0, and red is > 1000).

Table 17 Results on the specular PS dataset (see Fig. 14). All methods present a similar systematic bias due to outliers located on the specular points

Method	MSE (px)	SSIM	CPU (s)
FFT [17]	66.68	0.92	< **0.01**
DCT [18]	46.16	0.95	0.01
FM [15]	94.25	0.90	0.09
Sylvester [28]	928.69	0.55	0.30
FM-PCG	**40.48**	**0.96**	0.24

5 Conclusions and perspectives

We demonstrated the properties of the proposed FM-PCG surface normal integrator. It combines all the efficiency benefits of FM, Krylov-based and preconditioning components while retaining the robustness and accuracy of the underlying variational approach.

All of the desirable properties in Section 4, including especially the flexibility to handle non-trivial domains are met by the proposed method.

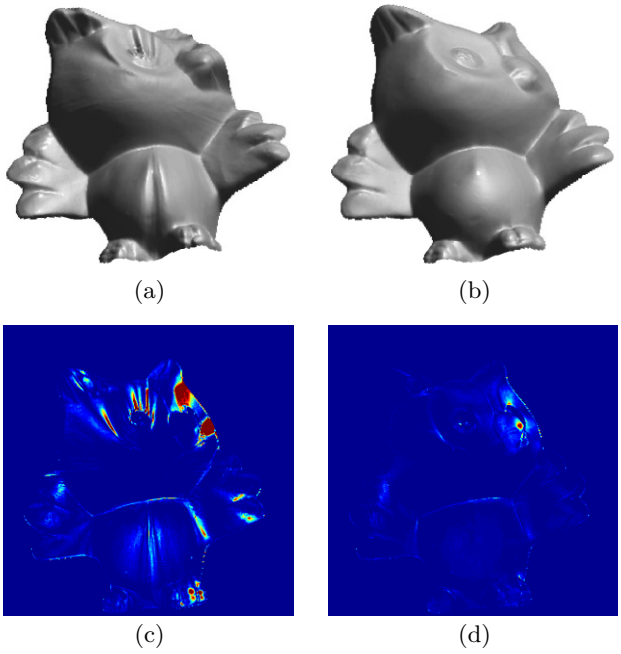


Fig. 15 Results of (a) improved FM and (b) improved FM-PCG methods introducing a smoothness constraint on the outliers. The corresponding MSE maps (c) and (d) show that errors due to the outliers are much reduced.

Table 18 Results of the improved methods on the same dataset as in Table 17. All MSE are significantly reduced

Method	MSE (px)	SSIM	CPU (s)
FFT [17]	17.26	0.95	< **0.01**
DCT [18]	14.79	**0.96**	0.01
FM [15]	34.47	0.91	0.09
Sylvester [28]	21.93	0.88	0.31
FM-PCG	**10.41**	**0.96**	0.27

It is clear that the proposed new integration scheme generates the most accurate reconstructions independently of the underlying conditions. The computational costs are very low and in most cases the method is faster than the recent Sylvester method of Harker and O'Leary. Only DCT is much faster, but DCT results are of low quality when the computational domain is not rectangular.

Therefore, the FM-PCG integrator is a good choice for applications which require accurate and robust 3D reconstruction at relatively low computational cost.

Nonetheless, our integration method remains limited to *smooth* surfaces. Studying the impact of appropriate preconditioning and initialisation on iterative methods which allow depth discontinuities, as for instance Refs. [9, 27], is an interesting problem. We are considering extending our study to *multi-view* normal field integration [54] to be an exciting avenue, which would allow the recovery of a full 3D shape, instead of a depth map.

References

[1] Pérez, P.; Gangnet, M.; Blake, A. Poisson image editing. *ACM Transactions on Graphics* Vol. 22, No. 3, 313–318, 2003.

[2] Horn, B. K. P.; Brooks, M. J. The variational approach to shape from shading. *Computer Vision, Graphics and Image Processing* Vol. 33, No. 2, 174–208, 1986.

[3] Woodham, R. J. Photometric method for determining surface orientation from multiple images. *Optical Engineering* Vol. 19, No. 1, 191139, 1980.

[4] Zafeiriou, S.; Atkinson, G. A.; Hansen, M. F.; Smith, W. A. P.; Argyriou, V.; Petrou, M.; Smith, M. L.; Smith, L. N. Face recognition and verification using photometric stereo: The photoface database and a comprehensive evaluation. *IEEE Transactions on Information Forensics and Security* Vol. 8, No. 1, 121–135, 2013.

[5] Smith, M. L.; Stamp, R. J. Automated inspection of textured ceramic tiles. *Computers in Industry* Vol. 43, No. 1, 73–82, 2000.

[6] Esteban, C. H.; Vogiatzis, G.; Cipolla, R. Multiview photometric stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 30, No. 3, 548–554, 2008.

[7] Haque, S. M.; Chatterjee, A.; Govindu, V. M. High quality photometric reconstruction using a depth camera. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2275–2282, 2014.

[8] Harker, M.; O'Leary, P. Least squares surface reconstruction from measured gradient fields. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1–7, 2008.

[9] Durou, J.-D.; Aujol, J.-F.; Courteille, F. Integrating the normal field of a surface in the presence of discontinuities. In: *Energy Minimization Methods in Computer Vision and Pattern Recognition*. Cremers, D.; Boykov, Y.; Blake, A.; Schmidt, F. R. Eds. Springer Berlin Heidelberg, 261–273, 2009.

[10] Klette, R.; Schlüns, K. Height data from gradient fields. In: Proceedings of SPIE 2908, Machine Vision Applications, Architectures, and Systems Integration V, 204–215, 1996.

[11] Coleman Jr., E. N.; Jain, R. Obtaining 3-dimensional shape of textured and specular surfaces using foursource photometry. *Computer Graphics and Image Processing* Vol. 18, No. 4, 309–328, 1982.

[12] Wu, Z.; Li, L. A line-integration based method for depth recovery from surface normals. *Computer Vision, Graphics and Image Processing* Vol. 43, No. 1, 53–66, 1988.

[13] Robles-Kelly, A.; Hancock, E. R. A graph-spectral method for surface height recovery. *Pattern Recognition* Vol. 38, No. 8, 1167–1186, 2005.

[14] Ho, J.; Lim, J.; Yang, M. H.; Kriegmann, D. Integrating surface normal vectors using fast marching method. In: *Computer Vision–ECCV 2006*. Leonardis, A.; Bischof, H.; Pinz, A. Eds. Springer Berlin Heidelberg, 239–250, 2006.

[15] Galliani, S.; Breuß, M.; Ju, Y. C. Fast and robust surface normal integration by a discrete eikonal equation. In: Proceedings of the 23rd British Machine Vision Conference, 2012.

[16] Bähr, M.; Breuß, M. An improved eikonal method for surface normal integration. In: *Pattern Recognition*. Gall, J.; Gehler, P.; Leibe, B. Eds. Springer International Publishing, 274–284, 2015.

[17] Frankot, R. T.; Chellappa, R. A method for enforcing integrability in shape from shading algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 10, No. 4, 439–451, 1988.

[18] Simchony, T.; Chellappa, R.; Shao, M. Direct analytical methods for solving Poisson equations in computer vision problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* Vol. 12, No. 5, 435–446, 1990.

[19] Wei, T.; Klette, R. A wavelet-based algorithm for height from gradients. In: *Robot Vision*. Klette, R.; Peleg, S.; Sommer, G. Eds. Springer Berlin Heidelberg, 84–90, 2001.

[20] Kovesi, P. Shapelets correlated with surface normals produce surfaces. In: Proceedings of the 10th IEEE International Conference on Computer Vision, Vol. 2, 994–1001, 2005.

[21] Wei, T.; Klette, R. Depth recovery from noisy gradient vector fields using regularization. In: *Computer Analysis of Images and Patterns*. Petkov, N.; Westenberg, M. A. Eds. Springer Berlin Heidelberg, 116–123, 2003.

[22] Karaçali, B.; Snyder, W. Noise reduction in surface reconstruction from a given gradient field. *International Journal on Computer Vision* Vol. 60, No. 1, 25–44, 2004.
[23] Agrawal, A.; Raskar, R.; Chellappa, R. What is the range of surface reconstructions from a gradient field? In: *Computer Vision–ECCV 2006*. Leonardis, A.; Bischof, H.; Pinz, A. Eds. Springer Berlin Heidelberg, 578–591, 2006.
[24] Badri, H.; Yahia, H. M.; Aboutajdine, D. Robust surface reconstruction via triple sparsity. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2283–2290, 2014.
[25] Du, Z.; Robles-Kelly, A.; Lu, F. Robust surface reconstruction from gradient field using the L1 norm. In: Proceedings of the 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications, 203–209, 2007.
[26] Reddy, D.; Agrawal, A. K.; Chellappa, R. Enforcing integrability by error correction using $l1$-minimization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2350–2357, 2009.
[27] Quéau, Y.; Durou, J.-D. Edge-preserving integration of a normal field: Weighted least squares, TV and L^1 approaches. In: *Scale Space and Variational Methods in Computer Vision*. Aujol, J.-F.; Nikolova, M.; Papadakis, N. Eds. Springer International Publishing 576–588, 2015.
[28] Harker, M.; O'Leary, P. Regularized reconstruction of a surface from its measured gradient field. *Journal of Mathematical Imaging and Vision* Vol. 51, No. 1, 46–70, 2015.
[29] Breuß, M.; Quéau, Y.; Bähr, M.; Durou, J.-D. Highly efficient surface normal integration. In: Proceedings of the 20th Conference on Scientific Computing, 204–213, 2016.
[30] Meister, A. 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* Vol. 140, No. 2, 311–345, 1998.
[31] Saad, Y. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2003.
[32] Durou, J.-D.; Courteille, F. Integration of a normal field without boundary condition. In: Proceedings of the 1st International Workshop on Photometric Analysis for Computer Vision, 2007.
[33] Kimmel, R.; Sethian, J. A. Optimal algorithm for shape from shading and path planning. *Journal of Mathematical Imaging and Vision* Vol. 14, No. 3, 237–244, 2001.
[34] Tsitsiklis, J. N. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control* Vol. 40, No. 9, 1528–1538, 1995.
[35] Sethian, J. A. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America* Vol. 93, No. 4, 1591–1595, 1996.
[36] Helmsen, J. J.; Puckett, E. G.; Colella, P.; Dorr, M. Two new methods for simulating photolithography development in 3D. In: Proceedings of SPIE 2726, Optical Microlithography IX, 253–261, 1996.
[37] Sethian, J. A. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.
[38] Yatziv, L.; Bartesaghi, A.; Sapiro, G. O(N) implementation of the fast marching algorithm. *Journal of Computational Physics* Vol. 212, No. 2, 393–399, 2006.
[39] Cacace, S.; Cristiani, E.; Falcone, M. Can local single-pass methods solve any stationary Hamilton–Jacobi–Bellman equation? *SIAM Journal on Scientific Computing* Vol. 36, No. 2, A570–A587, 2014.
[40] Zimmer, H.; Bruhn, A.; Valgaerts, L.; Breuß, M.; Weickert, J.; Rosenhahn, B.; Seidel, H.-P. PDE-based anisotropic disparity-driven stereo vision. In: Proceddings of the 13th International Fall Workshop Vision, Modeling, and Visualization, 263–272, 2008.
[41] Meister, A. *Numerik Linearer Gleichungssysteme. Eine Einführung in Moderne Verfahren*. Springer Spektrum, 2014.
[42] Hestenes, M. R.; Stiefel, E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards* Vol. 6, No. 49, 46–70, 1952.
[43] Meurant, G. *Computer Solution of Large Linear Systems*. Elsevier Science, 1999.
[44] Meurant, G. *The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations*. Society for Industrial and Applied Mathematics, 2006.
[45] Golub, G. H.; van Loan, C. F. *Matrix Computation*, 3rd edn. Johns Hopkins, 1996.
[46] Meijerink, J. A.; van der Vorst, H. A. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation* Vol. 31, No. 137, 148–162, 1977.
[47] Kershaw, D. S. The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations. *Journal of Computational Physics* Vol. 26, No. 1, 43–65, 1978.
[48] Benzi, M. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics* Vol. 182, No. 2, 418–477, 2002.
[49] Kaasschieter, E. F. Preconditioned conjugate gradients for solving singular systems. *Journal of Computational and Applied Mathematics* Vol. 24, Nos. 1–2, 265–275, 1988.
[50] Tang, J. M.; Vuik, C. Acceleration of preconditioned Krylov solvers for bubbly flow problems. In: *Parallel Processing and Applied Mathematics*. Wyrzykowski, R.; Dongarra, J.; Karczewski, K.; Wasniewski, J. Eds. Springer Berlin Heidelberg, 1323–1332, 2008.
[51] Manteuffel, T. A. An incomplete factorization technique for positive definite linear systems. *Mathematics of Computation* Vol. 34, No. 150, 473–497, 1980.

[52] Wang, Z.; Bovik, A. C.; Sheikh, H. R.; Simoncelli, E. P. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing* Vol. 13, No. 4, 600–612, 2004.
[53] Noakes, L.; Kozera, R. Nonlinearities and noise reduction in 3-source photometric stereo. *Journal of Mathematical Imaging and Vision* Vol. 18, No. 2, 119–127, 2003.
[54] Chang, J. Y.; Lee, K. M.; Lee, S. U. Multiview normal field integration using level set methods. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1–8, 2007.

Martin Bähr is a Ph.D. student in mathematics at the Brandenburg Technical University in Germany. He received his master degree in applied mathematics at the same university in 2013. Since 2013, he works in the applied mathematics group with a scientific focus on mathematical image processing. His research interests include partial differential equations and numerical methods for image processing and computer vision.

Michael Breuß received his doctorate degree in mathematics from the University of Hamburg in 2001, and the habilitation in mathematics from the Technical University in Brunswick in 2006. For several years, he had been a member of the mathematical image analysis group in Saarbrücken, Germany. Since 2016, he is professor for applied mathematics at the Brandenburg Technical University in Cottbus, Germany. His research interests are mainly in mathematical image processing and 3D vision, and include in particular numerical methods.

Yvain Quéau is a postdoctoral researcher at Technical University Munich. He received his Ph.D. degree in computer science from INP-ENSEEIHT, Université de Toulouse, in 2015. His research interests include 3D-reconstruction by photometric techniques (shape-from-shading and photometric stereo), as well as variational methods for solving computer vision and image processing problems.

Ali Sharifi Boroujerdi is a Ph.D. student at the Brandenburg Technical University in Germany. After being a bachelor of software engineering, he received his master degree in software engineering in 2013. His research interests include dynamic programming techniques as well as the field of artificial intelligence in general, especially deep learning, reinforcement learning, and big data analysis.

Jean-Denis Durou received his Ph.D. degree in computer science from the Université Paris Sud-Orsay in 1993, and the "Habilitation à Diriger les Recherches" from the Université Toulouse III-Paul Sabatier in 2007. He is an assistant professor at the Université Toulouse III since 1994, and a member of the VORTEX team at the IRIT Laboratory. His main research interest is 3D-vision. He is more specifically interested in photometric 3D-reconstruction, i.e., shape-from-shading and photometric stereo.