A Direct Numerical Approach to Perspective Shape-from-Shading

Oliver Vogel, Michael Breuß, Joachim Weickert

Mathematical Image Analysis Group, Faculty of Mathematics and Computer Science,
Saarland University, Building E1.1, 66041 Saarbrücken, Germany.
Email: {vogel,breuss,weickert}@mia.uni-saarland.de

Abstract

We introduce a novel numerical method for a recently developed perspective Shape-from-Shading model. In order to discretise the corresponding partial differential equation (PDE), Prados et al. employed the dynamical programming principle yielding a Hamilton-Jacobi-Bellman equation. We reduce that model to its essential, namely to the underlying Hamilton-Jacobi equation. For this PDE, we propose an efficient semi-implicit implementation. Numerical experiments show the usefulness of our approach: Besides reasonable computational times, the method is robust with respect to noise as well as to the choice of the numerical initial condition which is a delicate point for many SFS algorithms.

1 Introduction

The Shape-from-Shading (SFS) problem amounts to compute the 3-D shape of a surface from the brightness of exactly one given grey value image of that surface. It is a classical problem in computer vision, see e.g. [5, 6, 9, 21] and the references therein for an overview.

The modeling of the SFS problem via the use of a PDE was introduced by Horn [7, 8, 9], who also coined the name ‘shape-from-shading’. The model of Horn is the basis of all later works in that field. As it is of importance in the context of our work, let us mention some relevant features of the model of Horn. On the modeling side, a distinguished ingredient is the use of an orthographic camera, i.e., the camera performs an *orthographic projection* of the scene of interest. Together with a point light source at infinity, the PDE

$$|\nabla u| - \sqrt{\frac{1}{I^2} - 1} = 0 \qquad (1)$$

can be derived [1], where

- $u \equiv u(\mathbf{x})$ is the sought depth map,
- $|.|$ denotes the Euclidean vector norm,
- $I \equiv I(\mathbf{x}) = \frac{E(\mathbf{x})}{\sigma}$ is a normalised version of the image brightness, σ depends on the albedo of the surface and the intensity of the light source,
- $E \equiv E(\mathbf{x})$ is the brightness of the given grey-value image.

The *Eikonal equation* (1) constitutes the SFS-model widely studied in the literature. It is well-known that the corresponding problem is ill-posed, often shown via the so-called *convex-concave ambiguity*, see e.g. [4, 9] for related discussions.

In [2, 12, 13, 14, 15, 17, 18, 19, 20] a new PDE model for SFS is proposed. The setting of this new model is given by using a pinhole camera and a point light source at the optical center, thus incorporating a *perspective projection* instead of an orthographic one as in the classical case into the modeling process.

This perspective approach yields the *Hamilton-Jacobi equation*

$$\frac{I\mathsf{f}^2}{u}\sqrt{\frac{\mathsf{f}^2\,|\nabla u|^2 + (\nabla u \cdot \mathbf{x})^2}{Q^2} + u^2} = \frac{1}{u^2}\,, \qquad (2)$$

where $\mathbf{x} \in \mathbb{R}^2$ is in the image domain Ω as before, and

- f is the focal length relating the optical center of the camera and its retinal plane,
- $Q \equiv Q(\mathbf{x}) := \dfrac{\mathsf{f}}{\sqrt{|\mathbf{x}|^2 + \mathsf{f}^2}}$.

In order to obtain a viable numerical solver, Prados et al. use the dynamic programming principle. The resulting numerical solver incorporates the solution of an optimal control problem which can be quite intricate; for some details see [12]. In contrast, Tankus et al. rely on the level set method; see especially [20].

In a recent paper, Cristiani et al. [4] employed a semi-Lagrangian formulation of the above model to construct a fast SFS solver, however, as also shown in their paper, in contrast to our procedure their method is very sensitive to the choice of initial data.

Our contribution. We consider directly the PDE given in (2), and we show that it is possible to construct a *robust* and *efficent* numerical solver without the need to rely on dynamic programming, or to turn to the level set method. Another objective of us is that it is *easy to code* in comparison to other approaches in the field. As experiments show, the numerical routine we develop is also to a high degree *insensitive to perturbations* of initial data which can be a delicate point for SFS algorithms.

Organisation of this paper. In Section 2, we briefly review the modeling ingredients as well as some fundamental properties of the resulting PDE (2). In Section 3, we give a detailed description of our numerical scheme, focusing on its construction and the choice of the time step size. This discussion is followed by numerical experiments in Section 4. The paper is finished by concluding remarks.

2 Description of the model

In this paragraph, we briefly review the modeling process of (2), thereby illuminating the roles of its ingredients. For a more detailed description, see e.g. [15]. Figure 1 is adopted from there.

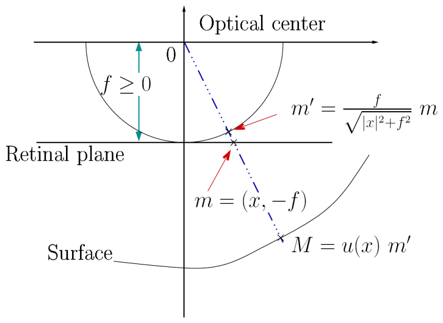

Figure 1: Perspective projection with a point light source located at the optical center.

Let Ω represent the rectangular image domain in $\mathbb{R}^2$. Consider then the surface $\mathcal{S}$, representing the object or scene of interest, parametrised by using the function $S : \bar{\Omega} \to R^3$ with

$$S(\mathbf{x}) = \frac{\mathsf{f}u(\mathbf{x})}{\sqrt{|\mathbf{x}|^2 + \mathsf{f}^2}} \left(\mathbf{x}, -\mathsf{f}\right)^T . \tag{3}$$

As the two columns of the Jacobian $\mathcal{J}[S(\mathbf{x})]$ are tangent vectors to $\mathcal{S}$ at the point $S(\mathbf{x})$, their cross-product is a normal vector to $\mathcal{S}$. Thus, a normal vector $\mathbf{n}(\mathbf{x})$ at the point $S(\mathbf{x})$ is given by

$$\begin{aligned} \mathbf{n}(\mathbf{x}) \quad = \quad & \Bigg(\mathsf{f}\nabla u(\mathbf{x}) - \frac{\mathsf{f}u(\mathbf{x})}{|\mathbf{x}|^2 + \mathsf{f}^2}\mathbf{x}\, , \\ & \nabla u(\mathbf{x}) \cdot \mathbf{x} + \frac{\mathsf{f}u(\mathbf{x})}{|\mathbf{x}|^2 + \mathsf{f}^2}\mathsf{f}\Bigg)^T . \end{aligned} \tag{4}$$

Assuming that the surface is Lambertian, the brightness equation is

$$I(\mathbf{x}) = \frac{\cos\theta}{r^2} . \tag{5}$$

Thereby, θ is the angle between the surface normal vector and the (unit) light source direction $\mathbf{L}$,

$$\mathbf{L}\left(S(\mathbf{x})\right) = \frac{1}{\sqrt{|\mathbf{x}|^2 + \mathsf{f}^2}} \left(-\mathbf{x}, \mathsf{f}\right)^T , \tag{6}$$

and r is the distance of the corresponding surface point to the light source. Employing the standard formula for cos in (5),

$$\cos\theta = \mathbf{L}\left(S(\mathbf{x})\right) \cdot \frac{\mathbf{n}(\mathbf{x})}{|\mathbf{n}(\mathbf{x})|} , \tag{7}$$

and evaluating the scalar product in (7), one obtains the sought formula (2).

For convenience, we assume for our numerical implementation, that the surface $\mathcal{S}$ is visible, i.e., it is in the front of the optical center, so that u is strictly positive. Then we use the change of variables $v = \ln(u)$, yielding the PDE

$$\frac{I\mathsf{f}^2}{Q} \sqrt{\mathsf{f}^2 \left|\nabla v\right|^2 + \left(\nabla v \cdot \mathbf{x}\right)^2 + Q^2} = e^{-2v} . \tag{8}$$

This equation is the basis of our numerical scheme.

Remarks. (i) As discussed in [12], the corresponding model was already considered in non-PDE-form in [10, 11]. (ii) The benefit of the above formulation is that the model is well-posed; for details see [12]. (iii) With different parametrizations of the surface $\mathcal{S}$, one arrives at different, yet to (8) equivalent PDEs.

The PDE (8) needs to be supported by boundary conditions, i.e., $v(\mathbf{x}) := \varphi(\mathbf{x})$ for $\mathbf{x} \in \partial\Omega$. In this setting, one can prove uniqueness of viscosity sub- and supersolutions, see especially [12].

We will be interested in computing a viscosity supersolution: this approach avoids the convex/concave ambiguity often encountered in SFS models. The corresponding theoretical setting can be described via $\varphi \equiv +\infty$; see the discussion in [12]. The consequence is, that the boundary condition becomes virtually unimportant, and we could implement it like Dirichlet boundary conditions given by a 'large' constant. In practice, we use Neumann boundary conditions in order to avoid the problem to set the latter, which works well. However, let us stress, that this particular consequence arises by an *Eulerian formulation* of the problem as given by (8). It does not hold, e.g., in the case of the recent semi-Lagrangian approach of Cristiani et al. [4]. We test the use of both types of boundary conditions in the section on numerical experiments.

3 Our numerical method

There are two main approaches in dealing with the PDE of interest (8):

1. Treat it like a usual boundary value problem and solve it directly.
2. Employ an additional 'time' variable and iterate until a steady state is reached.

We will follow the second path: This guarantees that we obtain an approximation of the viscosity supersolution. The logic is to choose an initial state above the supersolution and converge to this closest solution of (8) by iterating in 'time'. Thus, the PDE to discretise is

$$v_t = -\underbrace{\frac{I\mathrm{f}^2}{Q}\sqrt{\mathrm{f}^2\left|\nabla_{\mathbf{x}}v\right|^2 + \left(\nabla_{\mathbf{x}}v\cdot\mathbf{x}\right)^2 + Q^2}}_{=:A} + e^{-2v} \quad (9)$$

where $v \equiv v(\mathbf{x}, t)$ now, and where A is a useful abbreviation for later use.

3.1 Scheme construction

We employ the standard notation $v_{i,j}^n := v\left(ih_1, jh_2, n\tau\right)$, where h_1 and h_2 are the mesh widths and i and j the coordinates of the pixel (i,j) in $\mathbf{x}_1$- and $\mathbf{x}_2$-direction, respectively, and where τ is a time step size yet to be determined. The discretisation of $v_t(\mathbf{x}, t)$ we use is given by the *Euler forward* formula

$$v_t(\mathbf{x}, t)|_{(\mathbf{x},t)=(ih_1,jh_2,n\tau)} \approx \frac{v_{i,j}^{n+1} - v_{i,j}^n}{\tau}. \quad (10)$$

A *stable discretisation* of the spatial derivatives in $\nabla_{\mathbf{x}}v$ is given by the *upwinding* concept: As it is well-known for the kind of PDEs like (9), the use of central differences leads to artificial oscillations resulting in a blow-up of the numerical solution [16].

The upwinding concept boils down to use one-sided finite differences in the appropriate direction. These are determined by following the characteristics of the solution, thus realising wave-propagation in the physically correct direction. Following the derivation of Rouy and Tourin [16], one obtains

$$v_{\mathbf{x}_1}(\mathbf{x}, t)|_{(\mathbf{x},t)=(ih_1,jh_2,t)} \quad (11)$$
$$\approx \max\left(0, \frac{v_{i+1,j} - v_{i,j}}{h_1}, \frac{v_{i-1,j} - v_{i,j}}{h_1}\right),$$
$$v_{\mathbf{x}_2}(\mathbf{x}, t)|_{(\mathbf{x},t)=(ih_1,jh_2,t)} \quad (12)$$
$$\approx \max\left(0, \frac{v_{i,j+1} - v_{i,j}}{h_2}, \frac{v_{i,j-1} - v_{i,j}}{h_2}\right).$$

Note, that in (11)-(12), we have not specified the time level yet.

The reason for the latter is due to computational efficiency we would like to achieve. For this, we employ a *Gauß-Seidel-type idea* which works as follows. Let us introduce a linear numeration of pixels, i.e., we store the unknowns in a vector whose length is the total number of pixels. We then set the computational nodes in such a way, that we proceed according to the ordering in Figure 2.

1	2	3	...
$n_x + 1$	$n_x + 2$	$n_x + 3$	...
$2n_x + 1$	$2n_x + 2$	$2n_x + 3$	...
$3n_x + 1$	$3n_x + 2$	$3n_x + 3$	...
...	...	...	...

Figure 2: Pixel ordering for the Gauß-Seidel-type method.

Having a close look at formulae (11)-(12), we notice that the stencil of the method incorporates the data

$$\begin{array}{ccc} & v_{i,j+1} & \\ v_{i-1,j} & v_{i,j} & v_{i+1,j} \\ & v_{i,j-1} & \end{array}. \quad (13)$$

This means, at a pixel (i, j) and iterating through the pixel list as in Figure 2, we have already computed $v_{i,j+1}^{n+1}$ and $v_{i-1,j}^{n+1}$. Thus, for the computation of $v_{i,j}^{n+1}$, we can use these already updated values to achieve an accelerated convergence of our scheme.

To summarise, approximating spatial derivatives at time $t = n\tau$, the time levels within formulae (11)-(12) are set by us as

$$\max\left(0, \frac{v_{i+1,j}^{n} - v_{i,j}^{n}}{h_1}, \frac{v_{i-1,j}^{n+1} - v_{i,j}^{n}}{h_1}\right),$$
$$\max\left(0, \frac{v_{i,j+1}^{n+1} - v_{i,j}^{n}}{h_2}, \frac{v_{i,j-1}^{n} - v_{i,j}^{n}}{h_2}\right), \tag{14}$$

respectively. For clarity, let us stress once more, that the values from time level $(n+1)\tau$ in (14) are already *fixed* for the computation of $v_{i,j}^{n+1}$.

Turning to the discretisation of $I(\mathbf{x})$, $Q(\mathbf{x})$ and $\mathbf{x}$ in (9), we see that this issue amounts pixelwise to simple explicit term evaluations, so that there is no problem to deal with these terms.

We now turn, finally, to the source term e^{-2v} in (9). Source terms like this typically result in a very small time step size when evaluated *explicitly*, i.e., if at time level $t = n\tau$ we approximate it via $\exp\left(v_{i,j}^{n}\right)$. Especially, this results in iterates changing very slowly, leading to excessive computational times in reaching steady state solutions. Thus, we consider an *implicit* discretisation of it, writing

$$e^{-2v(\mathbf{x},t)}|_{(x,t)=(i,j,n\tau)} \approx e^{-2v_{i,j}^{n+1}}. \tag{15}$$

This component of our algorithm makes it necessary to employ in *each point* an iterative solver of the arising nonlinear equation: denoting by $\hat{A}$ the discretised version of term A from (9), we obtain by the Euler forward formula (10) pixelwise the update formula

$$v_{i,j}^{n+1} = v_{i,j}^{n} - \tau\hat{A} + \tau e^{-2v_{i,j}^{n+1}} \tag{16}$$

which has to be solved for $v_{i,j}^{n+1}$. We do this by employing the classical Newton-method, letting it iterate until convergence (which requires in practice three or four iterations). In this context, let us stress explicitly, that by (16) it does not become necessary to solve a nonlinear system of equations: the task amounts to solve *pixelwise* a quite harmless *one-dimensional* nonlinear equation, done efficently by the one-dimensional Newton-method.

To summarise, we propose a relatively simple-to-implement, semi-implicit method, where the source term is evaluated implicitly, and where already computed values are taken into account wherever possible accelerating convergence.

3.2 Choosing the time step size

We now discuss the most critical number that needs to be specified, i.e., the time step size τ. It is well-known, that implicitly discretised terms do not incorporate a restriction on the time step size. In fact, for a completely implicit scheme, where all data in (14) *and* (15) were from time level $(n+1)\tau$ there would theoretically not be a restriction on the allowed time step size at all. However, a completely implicit formulation results in a quite complicated nonlinear system of equations to solve numerically, which is contrary to the philosophy followed here to propose an efficient easy-to-code scheme. In our method, we discretise only the source term implicitly. Thus, the term in (15) does not impose a stability restriction, whereas the contribution due to $\hat{A}$ in (16) should imply a stability bound.

In practice, estimates for an upper bound on the time step size are often too restrictive for direct use if the underlying problem involves many nonlinearities. An automatic choice based on such an upper bound may thus result in unnecessarily long computational times. Nevertheless, it yields a good starting point for a user's choice. We now proceed in the line of these considerations, computing a reasonable candidate for an initial choice of the time step size. As indicated, we neglect for this the implicitly discretised source term.

A meaningful stability criterion for numerical methods for PDEs of the considered type is a *discrete maximum-minimum-principle*, i.e., ideally, the numerical solution of each time step shall not produce oscillations by over- or undershooting neighbouring data. This discrete stability criterion is closely related to the notion of viscosity solutions discussed in paragraph 2, as such viscosity solutions enforce the corresponding property on the level of the PDE-formulation [3].

Let us stress here, that it makes sense to establish a discrete minimum-maximum-principle neglecting in the corresponding computations the source term: This has the character of establishing a *necessary condition* for stability.

Let us have a close look at the terms of importance in the corresponding 'reduced form' of (16):

$$v_{i,j}^{n+1} = v_{i,j}^{n} - \tau \hat{A} . \tag{17}$$

Then the task arises to estimate

$$\begin{aligned} &\left|\tau \hat{A}\right| \\ &\overset{!}{\leq} \quad \max\left(\frac{\left|v_{i+1,j}^{n} - v_{i,j}^{n}\right|}{h_1}, \frac{\left|v_{i-1,j}^{n+1} - v_{i,j}^{n}\right|}{h_1}, \right. \\ &\qquad \left. \frac{\left|v_{i,j+1}^{n+1} - v_{i,j}^{n}\right|}{h_2}, \frac{\left|v_{i,j-1}^{n} - v_{i,j}^{n}\right|}{h_2} \right) . \end{aligned} \tag{18}$$

Let us note that, in (18), the values $v_{i-1,j}^{n+1}$, $v_{i,j+1}^{n+1}$ are already fixed so that it makes sense to incorporate these data from time level $(n+1)\tau$ into the computation.

For clarity, let us point out explicitly, that a discrete maximum-minimum-principle holds if (18) is satisfied: the right hand side amounts to the maximal absolute difference between $v_{i,j}^{n}$ and the other given data within the computational stencil. Thus, if (18) is met, the largest possible change due to an update yields the maximum or minimum of this set, respectively.

For abbreviation, let us now denote the quantity on the right hand side of (18) by δv. Employing the notation $\nabla \hat{v}$ for the discretisation of ∇v introduced in (11)-(12) and (14), we can compute the following estimates:

$$|\nabla \hat{v}|^2 \quad \leq \quad \left(\sqrt{2\delta v^2}\right)^2 \; = \; 2\delta v^2 \, , \tag{19}$$

$$(\nabla \hat{v} \cdot \mathbf{x})^2 \quad \leq \quad (2\hat{\mathbf{x}}\delta v)^2 \; = \; 4\hat{\mathbf{x}}^2 \delta v^2 \, , \tag{20}$$

where in (20) we have used

$$\hat{\mathbf{x}} := \max_{i,j} \left(|i|\, h_1,\, |j|\, h_2\right) . \tag{21}$$

Note, that $\hat{\mathbf{x}}$ is a finite number but it can be quite large.

Plugging (19)-(20) into $\hat{A}$ yields

$$\begin{aligned} \hat{A} \quad &\leq \quad \frac{I\mathsf{f}^2}{Q} \sqrt{\mathsf{f}^2 2\delta v^2 + 4\delta v^2 \hat{\mathbf{x}}^2 + Q^2} \\ &\leq \quad I\mathsf{f}\sqrt{\mathsf{f}^2 + \hat{\mathbf{x}}^2}\sqrt{2\mathsf{f}^2\delta v^2 + 4\hat{\mathbf{x}}^2\delta v^2 + Q^2} \, . \end{aligned} \tag{22}$$

Up to (22), all steps involve rigorous estimates. As we only seek an estimate for choosing τ here, we may now employ the following simplification. As Q is a number in $(0,1)$ generally small in comparison with the other arising terms – which also comprise a 'pessimistic' estimation – we may neglect Q^2, arriving at

$$\max \hat{A} \approx \delta v I \mathsf{f} \sqrt{\mathsf{f}^2 + \hat{\mathbf{x}}^2}\sqrt{2\mathsf{f}^2 + 4\hat{\mathbf{x}}^2} \, . \tag{23}$$

From (18) and (23), as $\left|\tau \hat{A}\right| \leq \delta v$ shall hold, we obtain after a few trivial manipulations the inequality

$$\tau < \frac{1}{2I\mathsf{f}\left(\mathsf{f}^2 + \hat{\mathbf{x}}^2\right)} \, . \tag{24}$$

In SFS computations, the number on the right hand side of (24) is often very small, in a typical setting of our experiments around 10^{-5} to 10^{-7}. As indicated, this number can be relaxed by some factor as the estimation is pessimistic. Let us also note, while the discrete maximum-minimum-principle is enforced, on the other hand the theoretical maximum of local updates is allowed. This means, the absolute size of the number does not matter as much as it seems.

4 Numerical Experiments

In this paragraph, we show several numerical experiments on synthetic images in order to assess the performance of our algorithm.

The pyramid experiment. This experiment is very useful for investigating and visualising the influence of initial and boundary conditions. The task is to reconstruct the pyramid-shaped surface shown in Figure 3. Figure 4 shows a photograph of this surface with $\sigma = 1000$, $\mathsf{f} = 251.6$, $h_1 = h_2 = 1$, 256×256 pixels, where σ denotes the factor determined by light source intensity and surface albedo, and h_1, h_2 are as before the pixel widths in $\mathbf{x}_1$- and $\mathbf{x}_2$-direction, respectively. The rendering was done by ray-tracing the surface.

Note, as the surface consists of four triangles, only these triangles can be hit in the ray-tracing process. Consequently, the surface normal at the top of the pyramid does not point towards the camera, resulting in a maximum grey value of 228 instead of 255. Hence, we cannot expect the sharp top of the pyramid to be reconstructed perfectly.

As noted in the introduction, at the image boundary we employ Neumann boundary conditions. The algorithm was initialised with the constant $v \equiv \log 0.2$, which is larger than the actual solution.

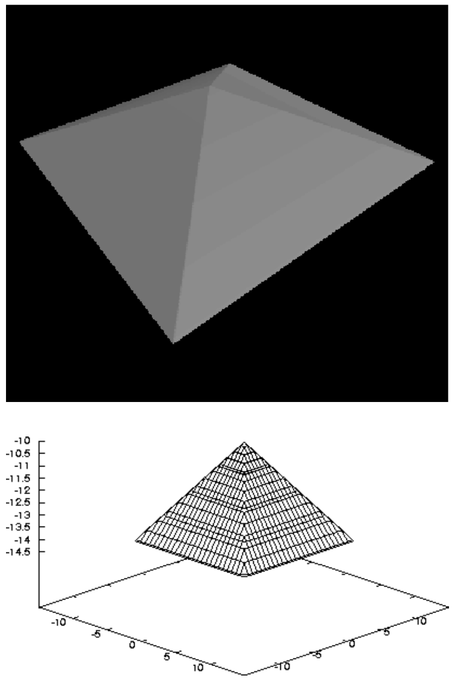

Figure 3: A pyramid shaped surface.

Let us point out here explicitly, that a constant v is equivalent to a spherical surface in the untransformed variable u. Figure 5 shows the surface corresponding to this constant initialisation.

Figure 6 shows a reconstruction of the surface using our method. As expected, the top of the pyramid is flat instead of sharp. Due to the boundary conditions, the reconstruction at the image boundary is a bit too round compared to the ground truth. However, overall the reconstruction of the pyramid is good, even the edges of the pyramid are reconstructed well.

As indicated before, our method does not rely on a specific initialisation. In the previous experiment, we initialised the algorithm with a constant v larger than the actual solution. We can choose any such constant, the reconstruction is always the same.

In fact, the method does not even need constant initialisation. Figure 7 shows an alternative initialisation for u with random data in the range $(0.18, 0.22)$. The result of the reconstruction is the same as shown in Figure 6.

Note that the speed of the reconstruction depends

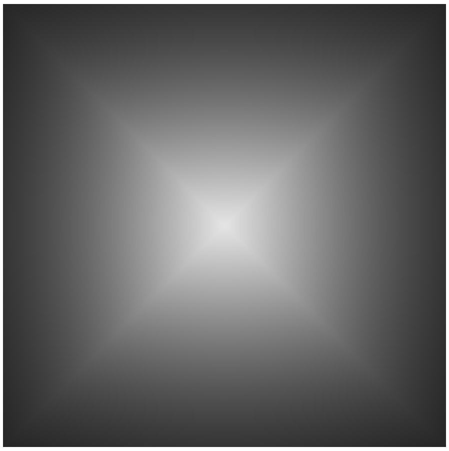

Figure 4: Input image for the pyramid surface.

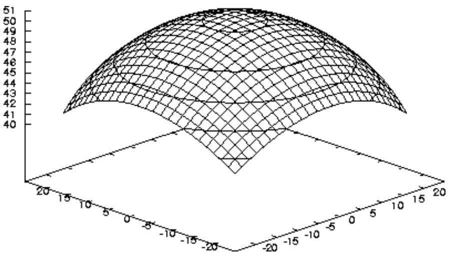

Figure 5: Surface with constant v.

on how far the initialisation is away from the actual solution and on the time step size. For ensuring stability, we need to use rather small time step sizes, see Section 3.2, so we should try to find an initialisation larger than, but as close as possible to the solution. In [12], it is shown that

$$v(\mathbf{x}) = -0.5 \log I f^2 \tag{25}$$

fulfils this. This initial image, however, is only defined if I is strictly positive everywhere, i.e., if there are no black pixels in the input image. With this initialisation, a good reconstruction of the pyramid image can be obtained in about one minute (C-code, Pentium 4, 3.2 GHz). Figure 8 shows the surface corresponding to this initialisation for the pyramid input image.

The discussed experiments were all done without any knowledge of the actual surface used within the

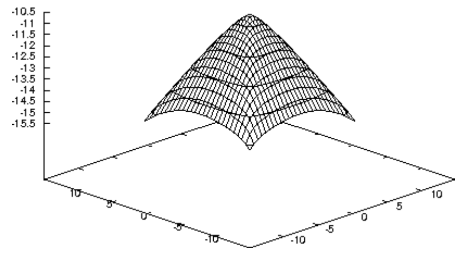

Figure 6: Reconstruction of the pyramid with Neumann boundary conditions.

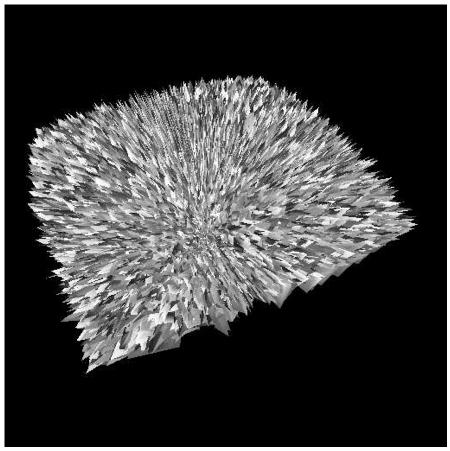

Figure 7: Surface with random v.

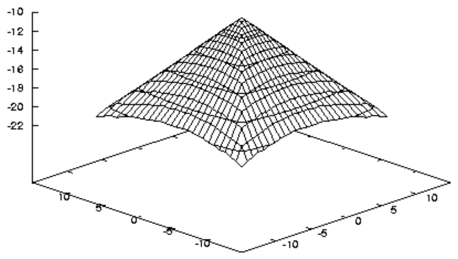

Figure 8: Initialisation with $v = -0.5 \log If^2$ for the pyramid image.

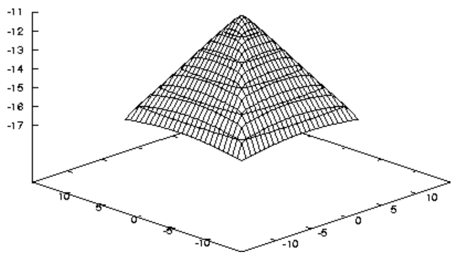

Figure 9: Reconstruction of the pyramid with exact Dirichlet boundary conditions.

algorithm. If we use exact Dirichlet boundary conditions, i.e., if we set the values at the image boundary to those of the ground truth, we obtain a nearly perfect reconstruction, which is shown in Figure 9.

Prados et al. [12] suggest to use state constraints boundary conditions, i.e., Dirichlet boundary conditions with a very large constant at the boundary. Since this means that very large gradients are generated at the boundary, it can easily be seen by taking into account (14), that state constraints boundary conditions are practically the same as Neumann boundary conditions (this is also true for the algorithm of Prados et al.).

We implemented the algorithm from [12]. Figure 10 shows a reconstruction obtained by using this algorithm. The result seems to be about as good as the one computed by our method.

Our algorithm is also robust with respect to noisy input data. In order to verify this claim, we have added Gaussian noise with standard deviation 10 to the input image. Figure 11 shows the corresponding reconstruction using our method. The result is nearly as good as without noise. The same holds true for the scheme of Prados et al.. Using the latter method, it is, however, necessary to reduce the time step size a bit to ensure stability of the method. The estimate given in [12] works very well for ”continuous” input data, but for ”discontinous” input data this estimate tends to be a bit too large, especially near very dark pixels, which may affect the stability of the scheme. Our method is perfectly stable using the estimate from equation (24).

Quantitative comparison with the scheme of Prados et al.. We compare our results to those of the algorithm of Prados et al. [12]. We compare both reconstruction quality and run time. For a quantitative comparison of both algorithms, we compare

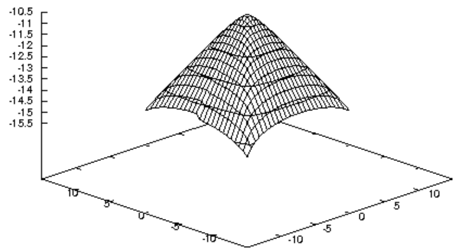

Figure 10: Reconstruction of the pyramid with the algorithm of Prados et al.

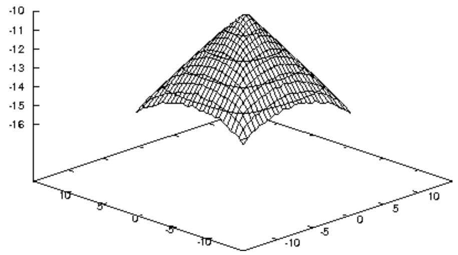

Figure 11: Reconstruction of the pyramid with Gaussian noise added.

the average L_1-error

$$\frac{1}{n_x n_y} \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} |u(i,j) - \hat{u}(i,j)| \,, \qquad (26)$$

where $n_x \times n_y$ is the image size, u the reconstructed depth and $\hat{u}$ the ground truth.

Table 1: L_1 errors of u for the pyramid experiment.

Noise	Our method	Prados et al.
Without	0.0069	0.0070
Gaussian, $\sigma = 5$	0.0071	0.0072
Gaussian, $\sigma = 10$	0.0076	0.0074

Table 1 shows the errors of both algorithms for the pyramid image with and without noise. Both algorithms are quite robust under noise, the results only get slightly worse. The quality of the result is about equal. As a stopping criterion we employed here a maximum change of a pixel value (in v) of less than 10^{-6}, which is fairly low, good results may as well be achieved with a larger value in less computation time.

Table 2: Run times for the pyramid experiment.

Method	Time
Prados et al.	322s
Our method, global τ	102s
Our method, local τ	41s

In Table 2, we compare the run times of both algorithms for the pyramid input image. Using a constant time step size according to the estimate in equation (24), we achieve a significantly better performance than Prados et al.. However, note that our implementation of the scheme in [12] is not optimal in the sense that it makes use of several (expensive) functions like atan, sin, cos, log, and exp in every iteration (while our algorithm only uses exp in the Newton step). One might speed it up, e.g. by using lookup tables for those functions. Still, our algorithm will be faster, and can be accellerated even more by evaluating the estimate from equation 24 in every pixel instead of using a constant time step size globally. For the pyramid image, we notice a speedup of roughly a factor 2.5 this way. Prados et al. make also use of a local choice of the time step size. All the run times in Table 2 are measured using a C-implementation of both methods on a Pentium 4, 3.2 GHz, 2 GB RAM running Linux, and both algorithms have been initialised like in equation (25).

Summary of the pyramid experiment. We have shown that our method works independently of the particular choice of initial and boundary values. In the simple setting of the pyramid experiment, the scheme yields good results. The new algorithm has proven robust to Gaussian noise added to the input image. Compared to the method of Prados et al., we observe a significant speedup with a comparable reconstruction quality.

The Mozart experiment. The Mozart experiment is a well-known benchmark in the SFS area. The ground truth is depicted in Figure 12.

As input image, we use the input image used in [12, 15], which is shown in Figure 13. Parameters for the reconstruction are $\mathsf{f} = 250$, $h_1 = h_2 = 1$, 256×256 pixels. Unfortunately, [12] does not give the value for σ, we just assume $\sigma = 4 \cdot 10^4$, which should be in a realistic range.

As a particular difficulty, the input data involve a slight 'break of the rules', as it does not satisfy the underlying modeling assumption that the face is completely visible, which is obvious by the shadows on its left and right hand side. These shadows are indeed black pixels in the image, hence $I(\mathbf{x})$ vanishes at these pixels. However, $I = 0$ also means our method will not move towards the solution, but remain at the initialisation values. To overcome this, we change all pixels in the input image with grey value smaller than 5 to a grey value equal to 5, this way, those pixels will also have the ability to move away from the initialisation. Nevertheless, the reconstruction of the Mozart image at these pixels is very difficult.

Figure 14 shows a reconstruction of the Mozart image using our method (with constant initialisation) and Neumann boundary conditions. The face and the shoulders of Mozart are reconstructed very well. The reconstruction is somewhat flat, but this is the case for nearly every reconstruction of the Mozart face. The (too high) reconstruction of the background makes the reconstruction appear even more flat.

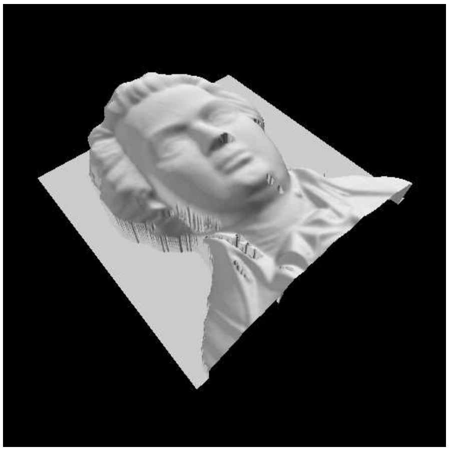
Figure 12: Ground truth for the Mozart image.

At the boundary of the face, the reconstruction proves to be very difficult, as we expected: Some (dark) pixels are far off the ground truth. The background is not recovered very well, this is caused by the boundary conditions (a sphere-like shape is assumed at the image boundary) and by the difficulties with the reconstruction of the face boundary.

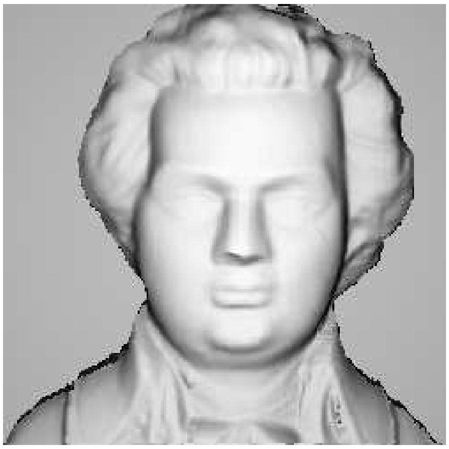
Figure 13: Mozart input image.

The reconstruction using the method of Prados et al. is again comparable to the one using our scheme, so we do not give an additional corresponding figure here. Since there are dark pixels in the input image, we need to use a smaller time step size once again to ensure convergence of the method of Prados et al..

Concluding the Mozart experiment. Comparing our result with those in [12] which can be considered as the state-of-the-art, the reconstruction quality of the face is similar. At the face boundary our method performs a bit better while we have a much smaller amount of outliers, yet the reconstruction of the background is comparable. Altogether, our method seems to do very well here, considering we had to guess σ.

5 Concluding remarks

To conclude, we have shown that the constructed numerical methods satisfies the intended goals:

- robustness of the scheme with respect to the choice of initial data,
- moreover, robustness with respect to the implementation of boundary conditions,
- robustness with respect to noisy data,
- numerical results are of the accuracy of state-of-the-art methods in the field, yet the numeri-

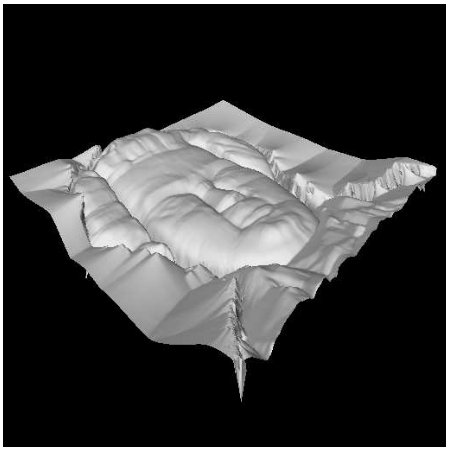

Figure 14: Reconstruction of the Mozart surface.

cal solver is much simpler,

- the implementation of the numerical scheme is of modest programming effort,
- computational times are reasonable.

Taking these aspects altogether, we have devised a highly competitive method in the field, as well as a good basis for further developments.

The work of us in the near future will be in two areas: (i) Further improvement and analysis of the numerical scheme, especially with respect to the choice of the time stepping method, and (ii) the investigation of real-world applications of the model in conjunction with the numerical scheme.

References

[1] A. R. Bruss. The eikonal equation: Some results applicable to computer vision. *Journal of Mathematical Physics*, 23(5):890–896, May 1982.

[2] F. Courteille, A. Crouzil, J.-D. Durou, and P. Gurdjos. Towards shape from shading under realistic photographic conditions. In *Proc. 2004 International Conference on Pattern Recognition*, volume 2, pages 277–280, Cambridge, MA, USA, August 2004.

[3] M. G. Crandall, H. Ishii, and P.-L. Lions. User's guide to viscosity solutions of second order partial differential equations. *Bulletin of the American Mathematical Society*, 27(1):1–67, 1992.

[4] E. Cristiani, M. Falcone, and A. Seghini. Some remarks on perspective shape-from-shading models. In F. Sgallari, A. Murli, and N. Paragios, editors, *Scale Space and Variational Methods in Computer Vision*, volume 4485 of *Lecture Notes in Computer Science*, pages 276–287, Berlin, May-June 2007. Springer.

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

[6] B. K. P. Horn. *Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View.* PhD thesis, Department of Electrical Engineering, MIT, Cambridge, MA, 1970.

[7] B. K. P. Horn. Understanding image intensities. *Artificial Intelligence*, 8:201–231, 1977.

[8] B. K. P. Horn and M. J. Brooks. Shape and source from shading. In *Proc. International Joint Conference on Artificial Intelligence*, pages 932–936, Los Angeles, CA, USA, August 1985.

[9] B. K. P. Horn and M. J. Brooks. *Shape from Shading*. Artificial Intelligence Series. MIT Press, Cambridge, MA, USA, 1989.

[10] T. Okatani and K. Deguchi. Reconstructing shape from shading with a point light source at the projection center: Shape reconstruction from an endoscope image. In *Proc. 1996 International Conference on Pattern Recognition*, volume 1, pages 830–834, Vienna, Austria, August 1996.

[11] T. Okatani and K. Deguchi. Shape reconstruction from an endoscope image by shape from shading technique for a point light source at the projection center. *Computer Vision and Image Understanding*, 66(2):119–131, May 1997.

[12] E. Prados. *Application of the theory of the viscosity solutions to the Shape from Shading problem*. PhD thesis, University of Nice Sophia-Antipolis, Oct. 2004.

[13] E. Prados and O. Faugeras. Perspective shape from shading and viscosity solutions. In *Proc Ninth International Conference on Computer Vision*, volume 2, pages 826–831, Nice, France, October 2003. IEEE Computer Society Press.

[14] E. Prados and O. Faugeras. A generic and provably convergent shape-from-shading method for orthographic and pinhole cameras. *International Journal of Computer Vision*, 65(1-2):97–125, 2005.

[15] E. Prados and O. Faugeras. Shape from shading: A well-posed problem? In *Proc. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 870–877, San Diego, CA, June 2005. IEEE Computer Society Press.

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

[17] A. Tankus, N. Sochen, and Y. Yeshurun. A new perspective [on] shape-from-shading. In *Proc. Ninth International Conference on Computer Vision*, volume 2, pages 862–869, Nice, France, Oct. 2003. IEEE Computer Society Press.

[18] A. Tankus, N. Sochen, and Y. Yeshurun. Perspective shape-from-shading by fast marching. In *Proc. 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 43–49, Washington DC, USA, June-July 2004. IEEE Computer Society Press.

[19] A. Tankus, N. Sochen, and Y. Yeshurun. Reconstruction of medical images by perspective shape-from-shading. In *Proc. 2004 International Conference on Pattern Recognition*, volume 3, pages 778–781, Cambridge, UK, Aug. 2004.

[20] A. Tankus, N. Sochen, and Y. Yeshurun. Shape-from-shading under perspective projection. *International Journal of Computer Vision*, 63(1):21–43, June 2005.

[21] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape from shading: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.