UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE COMPUTAÇÃO

Technical Report - IC-10-09 - Relatório Técnico

March - 2010 - Março

Depth from Slope by Weighted Multi-Scale Integration

Helena C. G. Leitão *Rafael F. V. Saracchini* *Jorge Stolfi*

Depth from Slope by Weighted Multi-Scale Integration

Helena C. G. Leitão (hcgl@ic.uff.br)
Rafael F. V. Saracchini (rsaracchini@ic.uff.br)
Jorge Stolfi (stolfi@ic.unicamp.br)

March 16, 2010

Abstract

We describe a robust method for recovery of the depth coordinate from a normal or slope map of a scene, such as those obtained through photometric stereo or interferometry. The key features of the method are multi-scale integration (with proper averaging and interpolation formulas) and the use of a reliability weight mask to handle regions with uncertain or unavailable slope data. We tested our algorithm with several slope maps computed from known height fields, and observed height errors of one pixel spacing or less. We have also tested it with success on slope maps of real objects obtained with photometric stereo methods.

Contents

1 Introduction

The *integration of a slope map* to yield a height map is a computational problem that arises in several computer vision contexts, such as shape-from-shading [3, 6] and multiple-light photometric stereo [4, 11]. Abstractly, in this problem we want to determine an unknown function $\mathbf{z}$ defined on some region D of $\mathbb{R}^2$, given its gradient $\operatorname{grad}\mathbf{z} = (\partial\mathbf{z}/\partial x, \partial\mathbf{z}/\partial y)$. That is, we wish to find $\mathbf{z}$ such that

$$\frac{\partial\mathbf{z}}{\partial x}(x,y) = \mathbf{f}(x,y) \tag{1}$$

$$\frac{\partial\mathbf{z}}{\partial y}(x,y) = \mathbf{g}(x,y) \tag{2}$$

at every point (x,y) within the region D, where $\mathbf{f}$ and $\mathbf{g}$ are two given real functions defined on D. It is well known that this problem has a differentiable solution if and only if

$$\frac{\partial\mathbf{f}}{\partial y}(x,y) = \frac{\partial\mathbf{g}}{\partial x}(x,y) \tag{3}$$

for all (x,y) in D. In that case the solution can be written in many ways. For a rectangular domain $D = [0 \mathbin{\_} m] \times [0 \mathbin{\_} n]$, for example, it can be written as

$$\mathbf{z}(x,y) = C + \int_0^y \mathbf{g}(0,v)\,dv + \int_0^x \mathbf{f}(u,y)\,du \tag{4}$$

where C is an arbitrary constant. (Note that the degree of freedom represented by C is an inherent feature of the original problem, not a limitation of the method.)

In practical contexts, there are at least three difficulties with this approach. First, the slope functions $\mathbf{f}$ and $\mathbf{g}$ are usually *discretized*, i.e. known only at certain *slope sampling points* $p[u,v]$, which usually form a regular orthogonal grid. Each sample value $f[u,v]$ (resp. g) is some weighted average of $\mathbf{f}$ (resp. $\mathbf{g}$) over some neighborhood of the sampling point $p[u,v]$. Note that photometric stereo methods will typically estimate the x-slope and the y-slope at the same points, e.g. at the center of each image pixel.

Second, the data is usually contaminated with *noise* arising from unavoidable measurement, quantization, and computation errors. At some points, the expected magnitude of the error may be so high that the slope is essentially unknown; and this may happen over large regions of the domain D. These regions include all pixels where the scene's surface is too dark for photometric stereo to work, shadowed, or outside the range of the illumination source.

Third, the height function $\mathbf{z}$ is usually *discontinuous.* The height field of a real scene almost always has cliff-like discontinuities at the silhouettes of solid objects. It may also have regions where the height function itself is poorly defined, e.g. where the scene is highly porous, covered with hair-like structures, or transparent.

Because of all these factors, equation (3) often fails, and simple solutions that are correct in theory may yield very poor results. In particular, replacing equation (4) by a discrete integral often results in very large jumps between successive scanlines, because the noise and discontinuities in the data will cause the integrals to drift away from the correct value, independently along each scanline. Moreover, equation (4) uses the y-slope information $\mathbf{g}(x, y)$ only at the left edge of the picture; so it throws away almost half of the data, and is extremely sensitive to the first column of $\mathbf{g}$. Most published solutions to this problem are attempts to use the redundancy of the data to reduce the effect of noise and discontinuities in it [8, 10, 7, 5].

2 Description of our algorithm

2.1 Laplacian equation

The first basic idea in our solution is to differentiate both sides of the equations (1) and (2) and add them to obtain a single functional equation

$$\mathbf{L}(\mathbf{z})(x, y) = \mathbf{D}(\mathbf{f}, \mathbf{g})(x, y) \tag{5}$$

where

$$\mathbf{L}(\mathbf{z}) = \frac{\partial^2 \mathbf{z}}{\partial x^2} + \frac{\partial^2 \mathbf{z}}{\partial y^2} \qquad \mathbf{D}(\mathbf{f}, \mathbf{g}) = \frac{\partial \mathbf{f}}{\partial x} + \frac{\partial \mathbf{g}}{\partial y} \tag{6}$$

In other words, the Laplacian $\mathbf{L}(\mathbf{z})$ of the height field $\mathbf{z}$ computed from $\mathbf{z}$ should be equal to the Laplacian $\mathbf{D}(\mathbf{f}, \mathbf{g})$ computed from the given slope fields $\mathbf{f}, \mathbf{g}$ [9, 6].

The advantage of this formulation is that equation (5) has a solution for *any* functions $\mathbf{f}, \mathbf{g}$. Actually equation (5) is under-determined because the homogeneous version

$$\mathbf{L}(\mathbf{z}) = 0 \tag{7}$$

is satisfied by any linear function of position $\mathbf{z}(x, y) = \alpha x + \beta y + \gamma$, which when added to any solution of equation (5) will yield infinitely many solutions. However, the space of solutions to (7) is relatively small, so these extra degrees of freedom can be removed by suitable "boundary conditions." These conditions can be derived from the slope data f and g in such a way as to use all valid information contained in the latter. We will defer this issue until after we describe our algorithm.

2.2 Discretizing the problem

The discrete nature of the data forces us to seek a discrete solution, that is, an array of height samples $z[u, v]$ nominally placed at *height sampling points* $q[u, v]$. The discrete version of equation (5) is a system of linear equations between finite difference operators applied to the elements of the arrays z, f, and g.

To obtain accurate and comparable derivative estimates for both sides of equation (5), we assume that the height sampling points $q[u, v]$ are displaced from the slope sampling points $p[u, v]$ by half a step in each direction. Specifically, we assume that $p[u, v]$ is the point $(u + 1/2, v + 1/2)$ of $\mathbb{R}^2$, while $q[u, v]$ is the point (u, v).

In other words, if the slope data has m samples in x and n samples in y, then we can assume that the domain of the problem is the rectangle $[0 \,\_\, nx] \times [0 \,\_\, ny]$ of $\mathbb{R}^2$, divided into m by n square cells or *pixels* of unit side. The slope sampling points $p[u, u]$ are the centers of the pixels, and each height sampling point $q[u,v]$ is the lower left corner of the pixel centered at $p[u,v]$. Note that the q array has $m+1$ columns and $n+1$ rows.

The Laplacian $\mathbf{L}(\mathbf{z})$ on the left side of equation (5) can be computed from the $\mathbf{z}$ values by the simple difference scheme

$$\mathcal{L}(z)[u,v] = z[u-1,v] + z[u+1,v] + z[u,v-1] + z[u,v+1] - 4z[u,v] \tag{8}$$

The value of $\mathcal{L}(z)[u,v]$ is an unbiased estimate for the Laplacian $\mathbf{L}(\mathbf{z})$ at the height sampling point $q[u,v]$. To obtain an estimate for the right-hand side $\mathbf{D}(\mathbf{f},\mathbf{g})$ at the same point, we use the formula

$$\begin{aligned} \mathcal{D}(f,g)[u,v] &= \frac{1}{2}(f[u,v-1] - f[u-1,v-1] + f[u,v] - f[u-1,v]) \\ &+ \frac{1}{2}(g[u-1,v] - g[u-1,v-1] + g[u,v] - g[u,v-1]) \end{aligned} \tag{9}$$

Note that the simple difference $f[u,v-1] - f[u-1,v-1]$ gives an estimate of $\partial\mathbf{f}/\partial x$ at the midpoint of between $q[u,v-1]$ and $q[u,v]$, while the difference $f[u,v] - f[u-1,v]$ gives an estimate for the midpoint between $q[u,v]$ and $q[u,v+1]$. By averaging the two differences we get an estimate valid for $q[u,v]$. A similar reasoning justifies the second part of formula (9).

Combining the two estimators, we get one linear equation

$$\mathcal{L}(z)[u,v] = \mathcal{D}(f,g)[u,v] \tag{10}$$

for every sample point $q[u,v]$ that has four neighbors, that is, does not lie on the boundary of the domain.

2.3 Height scale

The finite-difference equations (8) and (9) assume that the spacing between the slope sampling points is one unit of length. The heights obtained by solving the equations (10) will then be measured in pixels, too. Thus, for example, if $f[u,v] = \alpha$ and $g[u,v] = \beta$ for some constants α, β and for all u, v, then the solution of (10) should be $C + \alpha u + \beta v$.

2.4 Accounting for the weights

In the derivation of equation (10) we have assumed that all data values $f[u,v]$ and $g[u,v]$ are equally reliable, and we have ignored the boundary cases where some of the neighboring z, f and g values are missing. To account for boundaries and unreliable data, we will assume given also a *reliability weight* $w[u,v]$ for each slope sampling point $p[u,v]$. We assume that $w[u,v]$ is a non-negative number reflecting the relative trustworthiness of the corresponding slope values $f[u,v]$, $g[u,v]$.

If all the data is equally reliable, one should set $w[u,v] = 1$ for all u, v in the domain D. (As we shall see, only the relative magnitudes of the weights are important; multiplying all weights by a positive scale factor will not affect the result.) If the data for a point $[u,v]$ is completely unreliable or missing, one should set $w[u,v] = 0$. In particular, we will assume that $w[u,v]$ us zero for all pixels $[u,v]$ outside the domain D. The weights are often generated as a byproduct of the measurement process that yielded the slope data. The weight array may be a mask created by other criteria, e.g. one may set $w[u,v] = 0$ if $p[u,v]$ is known to be part of the background where the slope values are not known.

For example, the weights should be set to zero along silhouette lines, where the height field usually has an abrupt discontinuity of unknown magnitude. Without this precaution, the integrator will implicitly assume that the heights are continuous along those lines, and essentially glue the visible surfaces of the foreground and background objects, distoring both. For smooth objects, for example, one may assume that pixels where the slope is very high are adjacent to silhouette lines.

To make the formulas more readable, we introduce the following shorthands for weight and slope data adjacent to a specified height sampling point $q[u,v]$:

$$\begin{array}{lll} w_{--} = w[u-1,v-1] & f_{--} = f[u-1,v-1] & g_{--} = g[u-1,v-1] \\ w_{+-} = w[u,v-1] & f_{+-} = f[u,v-1] & g_{+-} = g[u,v-1] \\ w_{-+} = w[u-1,v] & f_{-+} = f[u-1,v] & g_{-+} = g[u-1,v] \\ w_{++} = w[u,v] & f_{++} = f[u,v] & g_{++} = g[u,v] \end{array} \tag{11}$$

except that each quantity in the left-hand side of these equations is set to 0 if the slope sampling point corresponding to the right-hand side lies outside the domain D.

We introduce also the shorthands for neighboring the four height values adjacent to a height value $z[u,v]$:

$$\begin{array}{ll} z_{x-} = z[u-1,v] & z_{y-} = z[u,v-1] \\ z_{x+} = z[u+1,v] & z_{y+} = z[u,v+1] \end{array} \tag{12}$$

To account for the weights, we first rewrite equations (8) and (9) to separate the contributions of each edge:

$$\mathcal{L}(z)[u,v] = (z_{x+} - z[u,v]) - (z[u,v] - z_{x-}) + (z_{y+} - z[u,v]) - (z[u,v] - z_{y-}) \tag{13}$$

$$\mathcal{D}(f,g)[u,v] = \frac{1}{2}(f_{+-} + f_{++}) - \frac{1}{2}(f_{--} + f_{-+}) + \frac{1}{2}(g_{-+} + g_{++}) - \frac{1}{2}(g_{--} + g_{+-}) \tag{14}$$

The first two terms in formulas (13) and (14) are estimatives for the derivative $\partial\mathbf{z}/\partial x$ on the midpoints of the two horizontal grid edges that are incident to $q[u,v]$. The next two terms are similar estimates of $\partial\mathbf{z}/\partial y$ on the midpoints of the two vertical edges incident to $q[u,v]$.

We then consider equation (10) as a linear combination of four equations

$$\begin{aligned} z[u,v] - z_{x+} &= -\frac{1}{2}(f_{+-} + f_{++}) \\ z[u,v] - z_{x-} &= +\frac{1}{2}(f_{--} + f_{-+}) \\ z[u,v] - z_{y+} &= -\frac{1}{2}(g_{-+} + g_{++}) \\ z[u,v] - z_{y-} &= +\frac{1}{2}(g_{--} + g_{+-}) \end{aligned} \tag{15}$$

Next we replace the simple averages in formula (14) and in the right-hand sides of formulas (15) by weighted averages, using the reliability weights:

$$\begin{aligned} z[u,v] - z_{x+} &= -\frac{w_{+-}f_{+-} + w_{++}f_{++}}{w_{+-} + w_{++}} \\ z[u,v] - z_{x-} &= +\frac{w_{--}f_{--} + w_{-+}f_{-+}}{w_{--} + w_{-+}} \\ z[u,v] - z_{y+} &= -\frac{w_{-+}g_{-+} + w_{++}g_{++}}{w_{-+} + w_{++}} \\ z[u,v] - z_{y-} &= +\frac{w_{--}g_{--} + w_{+-}g_{+-}}{w_{--} + w_{+-}} \end{aligned} \tag{16}$$

Finally, we produce a single equation by taking a weighted average of the four equations, using the *edge reliability weights*

$$\begin{aligned} w_{x-} &= w_{--} + w_{-+} \\ w_{x+} &= w_{+-} + w_{++} \\ w_{y-} &= w_{--} + w_{+-} \\ w_{y+} &= w_{-+} + w_{++} \end{aligned} \tag{17}$$

Note that the weights w_{x-}, w_{x+}, w_{y-}, and w_{y+} are zero if the corresponding grid edge lies outside the domain D, or if the slope data on both sides of the edge are missing. We also introduce the *total vertex weight*

$$\begin{aligned} w_t = w_t[u,v] &= w_{x-} + w_{x+} + w_{y-} + w_{y+} \\ &= 2(w_{--} + w_{+-} + w_{--} + w_{-+}) \end{aligned} \tag{18}$$

The final equation is

$$-\tilde{\mathcal{L}}(z)[u,v] = -\tilde{\mathcal{D}}(f,g)[u,v] \tag{19}$$

where

$$-\tilde{\mathcal{L}}(z)[u,v] = z[u,v] - \frac{w_{x-}}{w_t}z_{x-} - \frac{w_{x+}}{w_t}z_{x+} - \frac{w_{y-}}{w_t}z_{y-} - \frac{w_{y+}}{w_t}z_{y+} \tag{20}$$

and

$$\begin{aligned} -\tilde{\mathcal{D}}(f,g) &= \frac{w_{--}}{w_t}f_{--} + \frac{w_{-+}}{w_t}f_{-+} \\ &- \frac{w_{+-}}{w_t}f_{+-} - \frac{w_{++}}{w_t}f_{++} \\ &+ \frac{w_{--}}{w_t}g_{--} + \frac{w_{+-}}{w_t}g_{+-} \\ &- \frac{w_{-+}}{w_t}g_{-+} - \frac{w_{++}}{w_t}g_{++} \end{aligned} \tag{21}$$

The use of weights takes care automatically of the boundary cases along the margins of the domain D. Along the lower margin $v = 0$, for example, the final equation will relate $z[u,v]$ to its three neighbors z_{x-}, z_{y+} and z_{x+}, using only the four available data values f_{-+},g_{-+}, f_{++}, and g_{++}.

2.5 Uncoupled height values

The case when $w_t[u, v] = 0$ requires special treatment. This situation means that $z[u, v]$ is *uncoupled*, namely there is no information about the slopes surrounding $q[u, v]$. In that case, formulas (20) and (21) would result in invalid divisions 0/0. We fix this case by setting w_{--}, w_{-+} w_{+-} w_{++} to 1 whenever the corresponding pixel exists, and arbitrarily assuming $f_{\sigma\tau} = g_{\sigma\tau} = 0$ for all four pixels. This fix generally has the effect of setting $z[u, v]$ to an average of the neighboring z values, which is usually a convenient default value. In particular, for height sampling points $q[u, v]$ that are in the interior of D, the resulting equation is

$$z[u, v] - \frac{1}{4}z_{x-} - \frac{1}{4}z_{x+} - \frac{1}{4}z_{y-} - \frac{1}{4}z_{y+} = 0 \tag{22}$$

that sets $z[u, v]$ to the average of its four neighbors. Alternatively, one could use the equation $z[u, v] = 0$.

In any case, the values assigned to uncoupled pixels $z[u, v]$ will not affect any neighboring z value, unless that neighbor too is uncoupled. The value $\frac{1}{8}w_t[u, v]$ can be interpreted as a local reliability weight for $z[u, v]$: it is 1 if all four surrounding data pixels are entirely reliable, 0 if they are all missing, and between 0 and 1 in the remaining cases. Values of $z[u, v]$ that have $w_t[u, v] = 0$ should be considered meaningless in subsequent applications.

2.6 Assembling the system

In sections 2.2–2.5 we developed a discrete equation for each height sampling point $q[u, v]$. Note that there is a one-to-one correspondence between these equations and the height sampling points $q[u, v]$. Therefore the number of equations is equal to the number of unknowns.

We gather all these equations in a linear equation system

$$A\hat{\mathbf{z}} = b \tag{23}$$

where $\hat{\mathbf{z}}$ are the unknown heights z, arranged (in any order) as a vector with $N = (m + 1)(n + 1)$ elements; A is the $N \times N$ matrix of the coefficients in the left-hand sides of those equations; and b is the N-vectors containing the right-hand sides. Note that A does not depend on the given slopes, while b depends on them.

The system (23) is still partly indeterminate since the nonzero height map $h[x + (nx + 1)y] = z[x, y] = 1$ for all x, y satisfies the homogeneous system $A\hat{\mathbf{z}} = 0$.

3 Multiscale solver

The system (23) is usually too large to solve by direct elimination methods, therefore we use an iterative method (Gauss-Seidel or Jacobi). Note that the matrix A is large but quite sparse, with at most five non-zero terms in each row. Moreover, by using the same order for both equations and unknowns, we ensure that each equation has 1 as the diagonal coefficient, while the off-diagonal coefficients are non-positive and add to -1. These features ensure the convergence of the iterative solver. Furthermore, each equation has at most five nonzero coefficients in the left-hand side, so the system's coefficient matrix A uses only $O(N)$ storage, and the product $A\hat{\mathbf{z}}$ can be computed in $O(N)$ time

On the other hand, the convergence can be quite slow. If the initial guess differs by the correct solution only at one corner of the array z, the iterative method will propagate the adjustment to the rest of the array by a process mathematically similar to the diffusion of heat in a plate. The number of iterations needed to reduce the error to (say) one half its original magnitude, over the whole image, is at least proportional to the square of the domain's diameter, namely $m^2 + n^2$.

In order to get around this obstacle, we use a *multiscale* technique. Namely, to obtain the initial guess for the solver, we reduce the slope arrays f, g to half their original width and height, compute from this reduced-scale data a reduced-scale height map z, and expand the latter to the full scale. The reduced problem is solved recursively in the same way, until the array dimensions m and n are so small that the system can be efficiently solved.

In other words, we construct a sequence of slope maps $f^{(k)}$, where $f^{(0)} = f$ and each map $f^{(k+1)}$ is a reduced copy of the previous one $f^{(k)}$; and ditto for g. Let $f^{(m)}$ and $g^{(m)}$ be the smallest of these arrays. We integrate these slopes (e.g. by Gauss-Seidel or even Gaussian elimination) and obtain a height map $z^{(m)}$ at the same scale. Then we integrate the maps $f^{(k)}$ and $g^{(k)}$, in order of decreasing k, each time with an iterative method using the solution $z^{(k+1)}$ as the initial guess, and obtaining a more detailed solution $z^{(k)}$. The height map $z^{(0)}$ is the result.

If the reduction and expansion are done with a minimum amount of care, the recursively computed initial guess will be close to the correct solution in the broad features, so that the iterative solver will need to make only local adjustments (introducing details at a scale of one or two pixels).

3.1 The algorithm

The central part of our multiscale integration algorithm is the recursive procedure *ComputeHeights* whose pseudocode is given in figure 1. The procedure takes as inputs the slope maps *f*, *g* and the weight map *w*, and outputs a height map *z*. The slope maps and the weight map must have the same number of rows and the same number of columns. The computed height map will have one extra column and one extra row, as detailed in section 2.2. The parameter *minsize* can be set to

3.2 Reducing the slope maps

The slope maps are reduced by averaging the slopes in each block of 2×2 pixels. If the original data arrays have m columns and n rows, the reduced data arrays will have $\lceil m/2 \rceil$ columns and $\lceil n/2 \rceil$ rows.

Our algorithm takes the weights into account in the data reduction. Namely, when computing the X slope array $f^{(k+1)}$ from $f^{(k)}$, we set $f^{(k+1)}[u, v]$ to the *weighted* average of the four pixels $f^{(k)}[2u + \mu, 2v + \nu]$, with weights $w^{(k)}[2u + \mu, 2v + \nu]$, where μ, ν are either 0 or 1. The array $g^{(k+1)}$ is computed in the same way from $g^{(k)}$. In particular, if all four weights are zero, we just set $f^{(k+1)}[u, v] \leftarrow g^{(k+1)}[u, v] \leftarrow w^{(k+1)}[u, v] \leftarrow 0$.

```
$ComputeHeights(f, g, w)$
  1. $nx \leftarrow f.\mathrm{NX}$; $ny \leftarrow f.\mathrm{NY}$
  2. If $nx < minsize$ and $ny < minsize$ then
       3. $z \leftarrow (0, 0, \ldots, 0)$;
  4. else
       5. $f' \leftarrow ReduceSlopeMap(f, w)$;
       6. $g' \leftarrow ReduceSlopeMap(g, w)$;
       7. $w' \leftarrow ReduceWeightMap(w)$;
       8. $z' \leftarrow ComputeHeights(f', g', w')$;
       9. $z \leftarrow ExpandHeightMap(z')$;
  10. $A, b \leftarrow BuildSystem(f, g, w)$;
  11. $z \leftarrow SolveSystem(A, b, z)$;
  12. Return $z$.
```

Figure 1: The main procedure of the integrator.

3.3 Reducing the weight maps

For the multiscale algorithm we also need to compute a reliability weight array $w^{(k+1)}$ for the reduced slope arrays. We setting $w^{(k+1)}[u, v]$ to the harmonic mean of the four weights $w^{(k)}[2u + \mu, 2v + \nu]$. The harmonic mean tends to be closer to the minimum of the given values, and is zero if any of these values is zero.Therefore, any pixel of the reduced map that includes an original pixel with unreliable slope data will itself be marked unreliable. (A run-time option in our implementation allows the use of arithmetic mean instead of harmonic mean, if desired.)

As before, we assume that the weight $w^{(k)}[2u + \mu, 2v + \nu]$ is zero if the corresponding pixel lies outside the domain. Therefore, if the number of columns is odd, the last column is implicitly duplicated; and ditto if the number of rows is odd. This detail is not critical since it affects the reduced-scale solution only along the corresponding edges of the domain. Any such errors will be quickly corrected by the iterative solver at the original scale.

This reduction method implicitly assumes that each slope datum f_{++} or g_{++} is the average of the corresponding derivative $\partial\mathbf{z}/\partial x$ or $\partial\mathbf{z}/\partial y$ inside the grid cell centered at $p[u, v]$, with uniform weight. In practice, each datum will be a fuzzy average, with a smooth (e.g. Gaussian) weight kernel centered at that point. In that case, the reduced-scale slopes should be computed by a weighted average that includes some additional values outside that 2×2 block. However, here too the consequences of this improper averaging are small-scale errors that are quickly corrected at the original finer scale.

3.4 Expanding the height map

The height map computed at a given scale is expanded to the next-finer scale by simple linear interpolation and doubling of the values. Namely,

$$
\begin{aligned}
z^{(k-1)}[2u, 2v] &= 2z^{(k)}[u, v] \\
z^{(k-1)}[2u+1, 2v] &= 2\frac{1}{2}(z^{(k)}[u, v] + z^{(k)}[u+1, v]) \\
z^{(k-1)}[2u, 2v+1] &= 2\frac{1}{2}(z^{(k)}[u, v] + z^{(k)}[u, v+1]) \\
z^{(k-1)}[2u+1, 2v+1] &= 2\frac{1}{4}(z^{(k)}[u, v] + z^{(k)}[u+1, v] + z^{(k)}[u, v+1] + z^{(k)}[u+1, v+1])
\end{aligned}
\tag{24}
$$

for all integers u, v where the left-hand side is defined. Note that this procedure is well-defined even if the height map $z^{(k)}$ has odd dimensions, since the dimensions of $z^{(k+1)}$ are rounded up when building the reduced-scale problem. The factor of 2 is needed to account for the fact that the pixels of $z^{(k+1)}$ are twice as wide and tall as those of $z^{(k)}$.

3.5 The indeterminate linear term

As we observed in section 2.1, the solution to the basic equation (5) includes an arbitrary linear term $\alpha x + \beta y + \gamma$. In a single-scale solution, this term becomes determined in part by the slope values at the effective domain boundaries (the boundaries of D as well as the points adjacent to zero-weight pixels.) At these points the weighted Laplacians $\tilde{\mathcal{L}}(z)$ and $\tilde{\mathcal{D}}(f, z)$ become a combination of first- and second-order derivatives, so the equations become sensitive to slopes rather than just curvature. If the iteration is stopped early, the linear term is also determined in part by the average linear term contained in the starting guess.

Thus, in a single-scale solution one should post-process the computed height field z in order to fix the linear term. One could, for instance, compute the average X slope over D from the data $f[u, v]$ and from the height field z, and then add to the latter a term αx such that the two averages will become equal. Ditto for the Y slopes.

However, it turns out that multiscale integration usually gives a height field with correct linear terms, so that the above adjustment is not necessary. To understand why, observe that the coarsest-scale slope map has only one pixel, and and its value is the average slope over the whole domain D. Therefore the solution at that stage is basically the correct linear term, as determined by the average slopes. This overall linear term is preserved by the expansion of the height maps and by the iterative solvers at increasingly finer scales.

4 Tests

4.1 Synthetic test data

4.2 Height functions

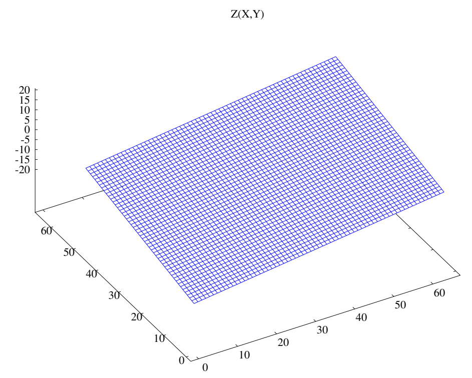
To test the integrator we used several slope maps computed from known height fields, shown in figures 2(a) through 9(a). Each height field was defined by an algebraic function $\mathbf{z}(x, y)$ of the continuous domain coordinates x and y. This function and its derivatives were discretized to a pixel grid with 64 columns and 48 rows. The functions are:

Function	Description
03	Linear ramp
04	Quadratic hump
05	Hemispherical dome
06	Five-sided pyramid
07	Conical mound
08	Concentric ripples
09	Babel tower
10	Divergent parabolic ramps

All functions are continuous, except function 10 which has a cliff-like discontinuity along a straight line starting near the center of the domain. Functions 05–07 have have discontinuous first derivatives, and function 08 has small-scale detail near the center.

In all test data, each reference height $z^*[u, v]$ was obtained by averaging the value of the function $\mathbf{z}(x, y)$ evaluated at 25 sampling points inside a one-pixel square centered at $q[u, v]$. The reliability weight maps w are discussed in section 4.2.3.

(a)



(b)

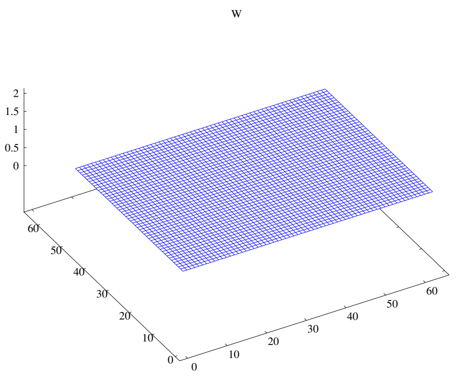


Figure 2: Height function 03 (linear ramp): original height map z^* (a) and input reliability weight map w (b).

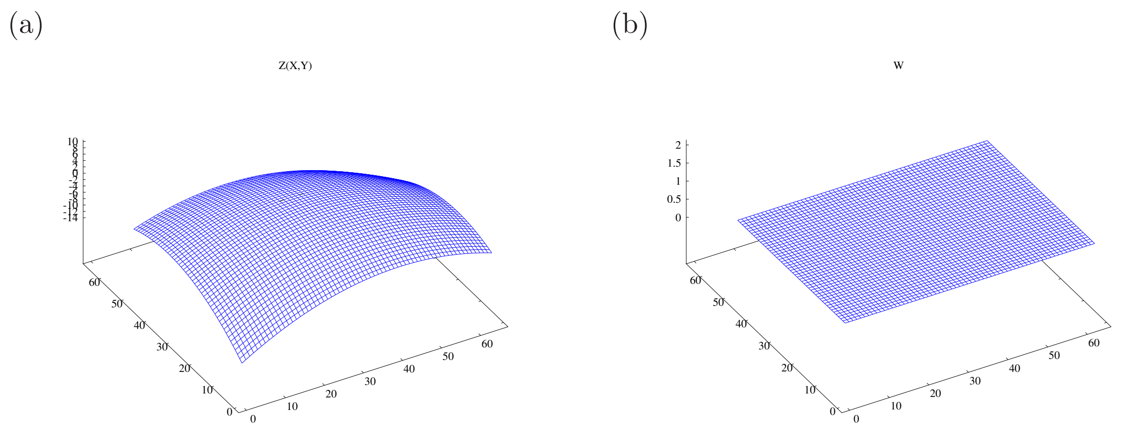


Figure 3: Height function 04 (quadratic hump): original height map z^* (a) and input reliability weight map w (b).

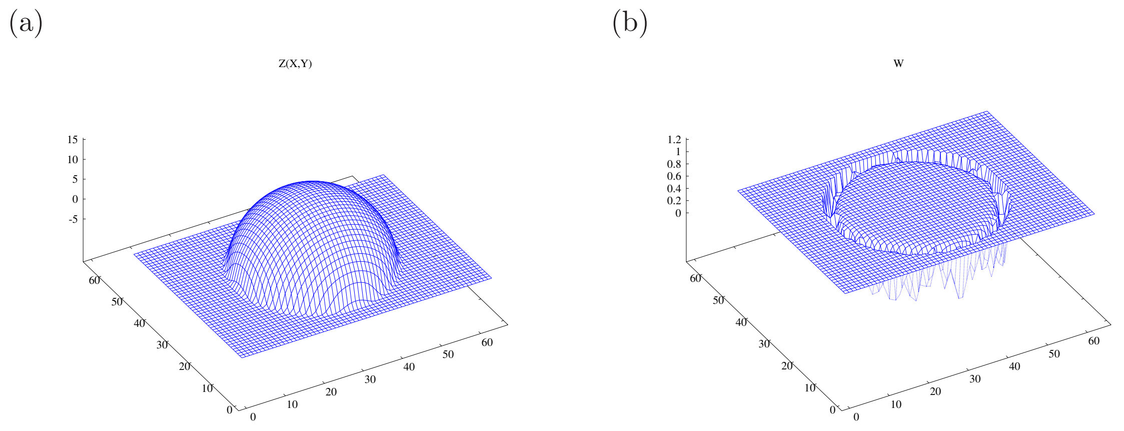


Figure 4: Height function 05 (hemispherical dome): original height map z^* (a) and input reliability weight map w (b).

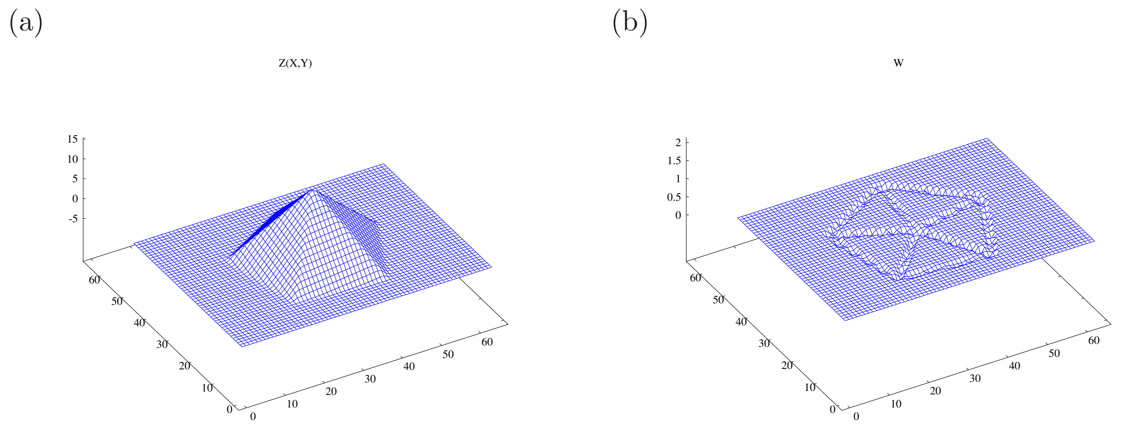
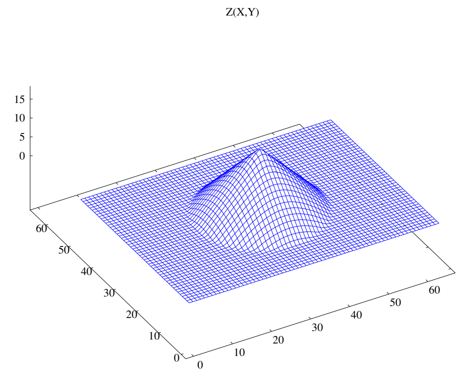


Figure 5: Height function 06 (five-sided pyramid): original height map z^* (a) and input reliability weight map w (b).

(a)



(b)

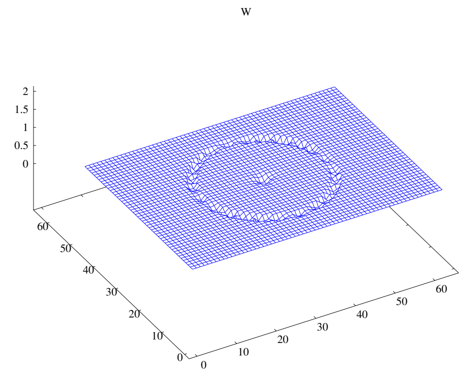
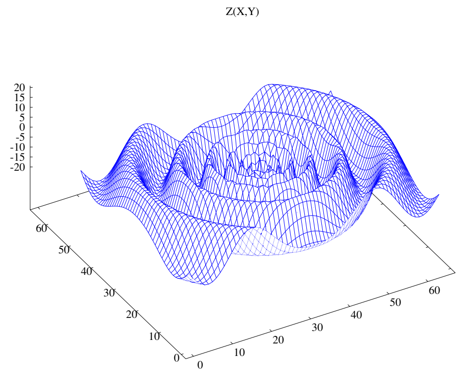


Figure 6: Height function 07 (conical mound): original height map z^* (a) and input reliability weight map w (b).

(a)



(b)

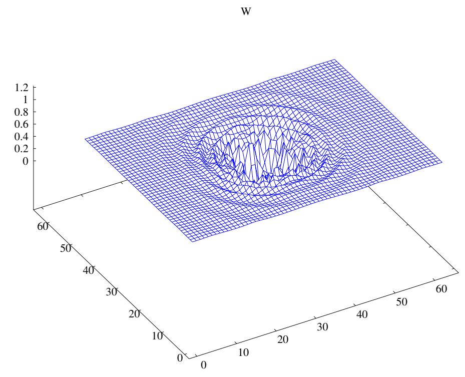
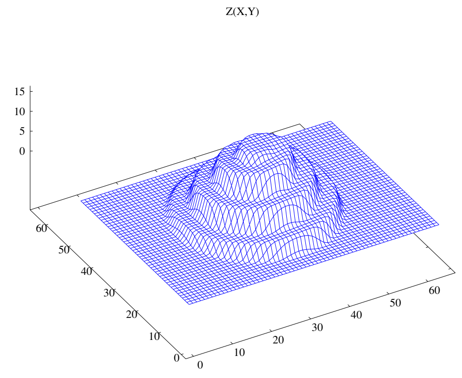


Figure 7: Height function 08 (concentric ripples): original height map z^* (a) and input reliability weight map w (b).

(a)



(b)

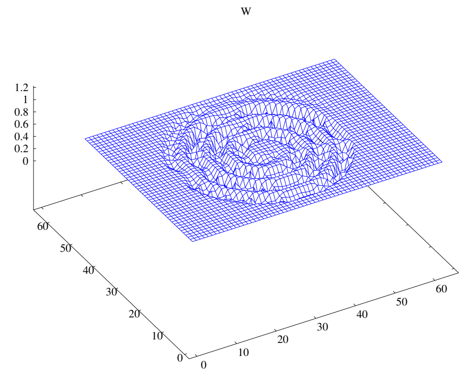
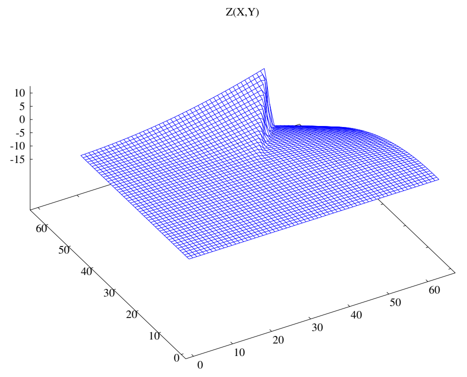


Figure 8: Height function 09 (Babel tower): original height map z^* (a) and input reliability weight map w (b).

(a)



(b)

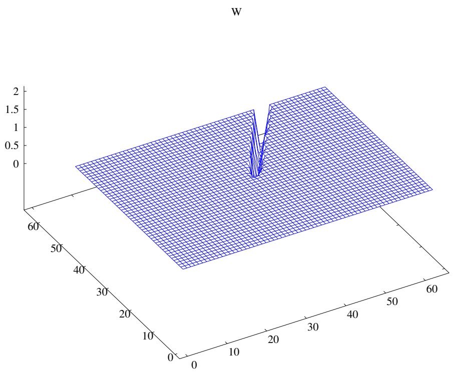
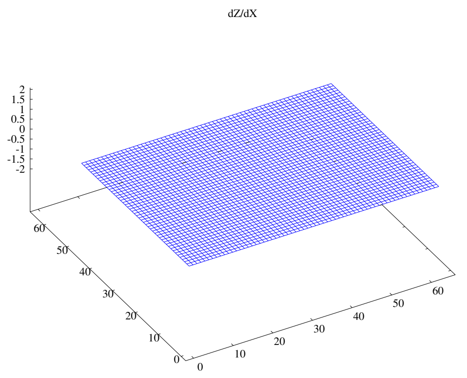


Figure 9: Height function 10 (divergent parabolic ramps): original height map z^* (a) and input reliability weight map w (b).

4.2.1 Numerical slopes

For each function, we obtained two sets of slope data $f[u, v], g[u, v]$, in two different ways. One set, the *numerical slopes*, was obtained by simple finite differences from the reference height map z^*. Namely, $f[u, v]$ was the average of $z^*[u+1, v] - z^*[u, v]$ and $z^*[u+1, v+1] - z^*[u, v+1]$; and similarly for $g[u, v]$. The maps are shown in figures 10 through 17. These slope maps are not representative of real input data, but serve to check the basic numerical correctness of the method.

(c)



(d)

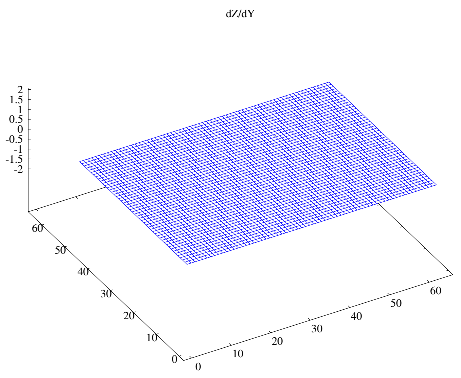
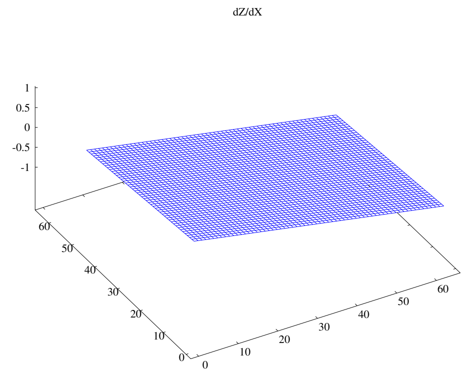


Figure 10: Numerical slope data for height function 03 (linear ramp): X derivative f (a) and Y derivative g (b).

(c)



(d)

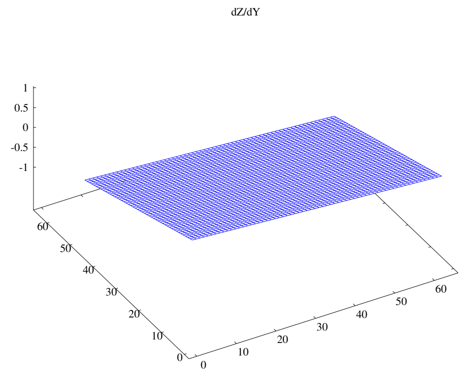
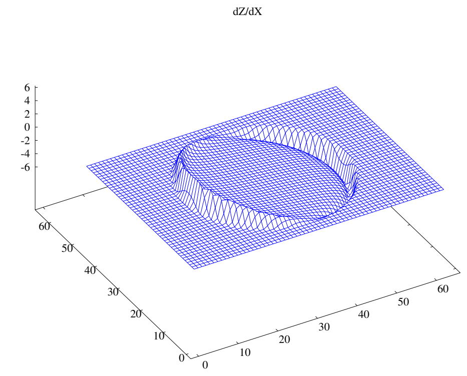


Figure 11: Numerical slope data for height function 04 (quadratic hump): X derivative f (a) and Y derivative g (b).

(c)



(d)

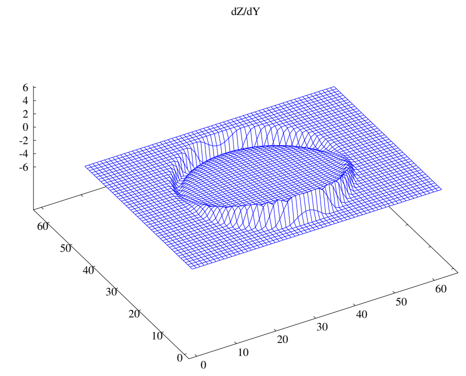
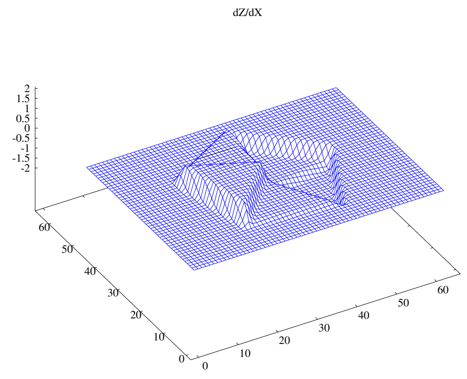


Figure 12: Numerical slope data for height function 05 (hemispherical dome): X derivative f (a) and Y derivative g (b).

(c)



(d)

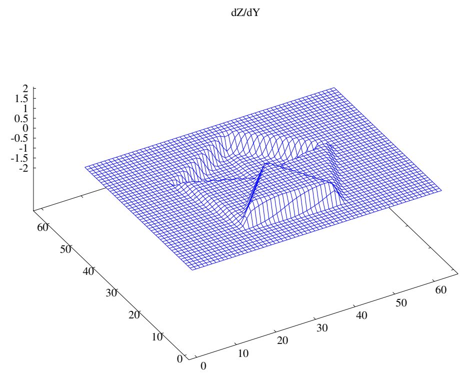
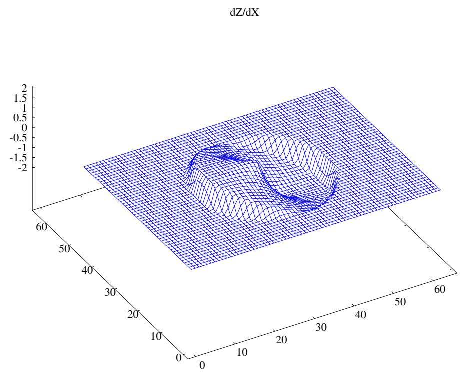


Figure 13: Numerical slope data for height function 06 (five-sided pyramid): X derivative f (a) and Y derivative g (b).

(c)



(d)

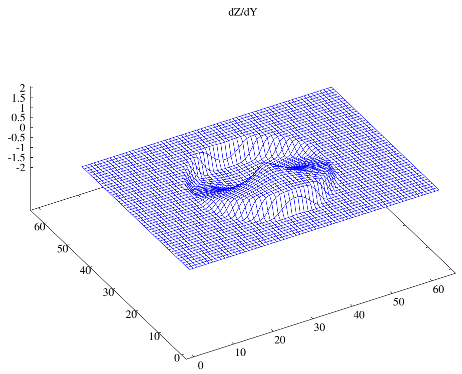
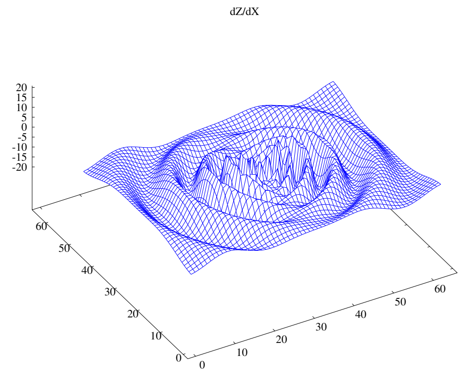


Figure 14: Numerical slope data for height function 07 (conical mound): X derivative f (a) and Y derivative g (b).

(c)



(d)

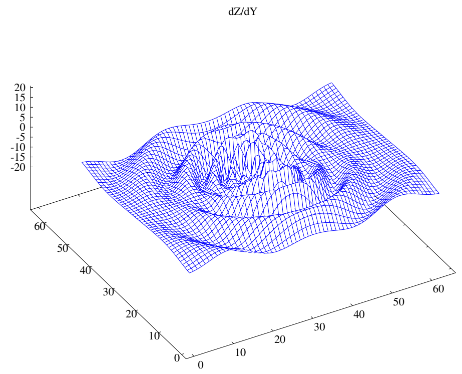
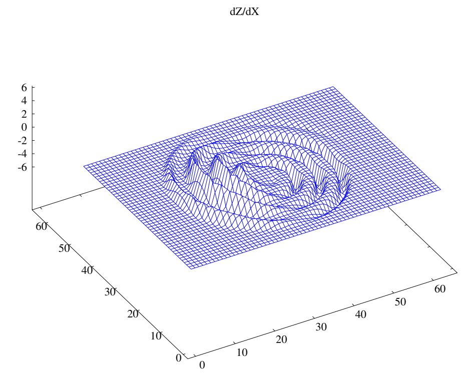


Figure 15: Numerical slope data for height function 08 (concentric ripples): X derivative f (a) and Y derivative g (b).

(c)



(d)

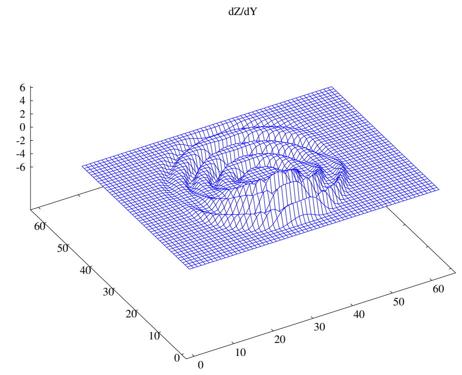
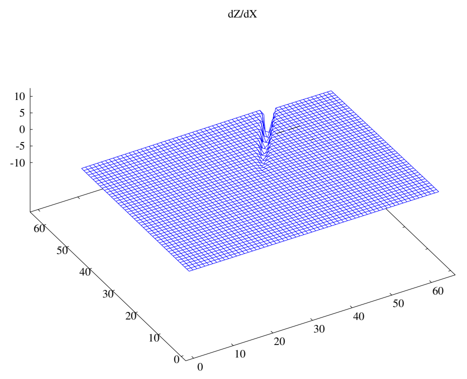


Figure 16: Numerical slope data for height function 09 (Babel tower): X derivative f (a) and Y derivative g (b).

(c)



(d)

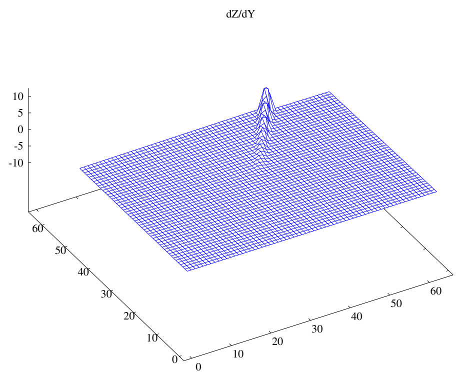


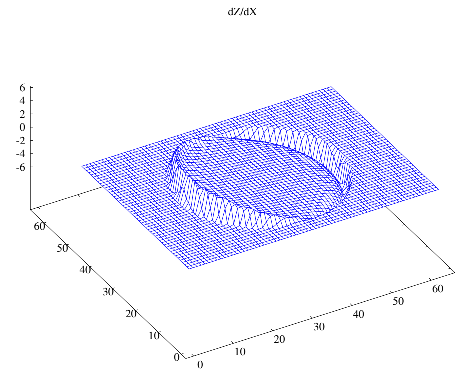
Figure 17: Numerical slope data for height function 10 (divergent parabolic ramps): X derivative f (a) and Y derivative g (b).

4.2.2 Analytic slopes

For each function, we also prepared a second input dataset, the *analytic slopes*, obtained by sampling the analyic gradient of the height function $\mathbf{z}(x, y)$. Specifically, each slope value $f[u, v]$ was set to the Hann-weighted average of the analyitc derivative $\partial\mathbf{z}/\partial x$ evaluated at 25 sampling points inside a 2×2 pixel window concentric with pixel $[u, v]$; and similarly for $g[u, v]$.

The analytic slope maps are shown in figures 18 through 22. Note that the analytic slopes can be quite misleading if the height function is discontinuous. For function 10, in particular, the analytic derivative of $\mathbf{z}$ in the direction perpendicular to the discontinuity line is zero everywhere, implying that the discontinuity does not exist. However, the analytic datasets are more representative of real data obtained by photometric stereo methods, which are generally sensitive to mean slope and insensitive to sharp height discontinuities. Except for function 10, the differences between the numerical and analytic slopes are rather subtle and largely confined to the lines of first-order discontinuity. For functions 03 (linear ramp) and 04 (quadratic hump), in particular, the analytic slopes are essentially identical to the numeric slopes, and their plots have been omitted to save space.

(c)



(d)

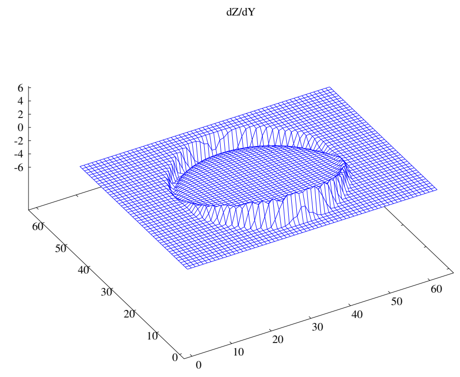
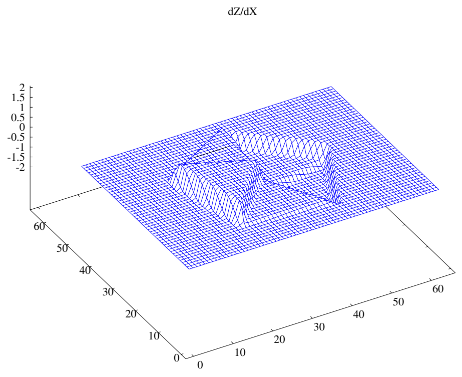


Figure 18: Analytic slope data for height function 05 (hemispherical dome): X derivative f (a) and Y derivative g (b). Compare with figure 12.

(c)



(d)

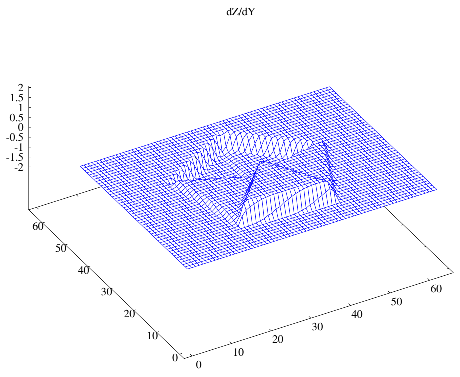
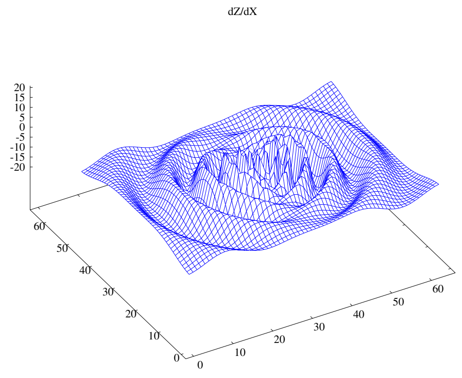


Figure 19: Analytic slope data for height function 06 (five-sided pyramid): X derivative f (a) and Y derivative g (b). Compare with figure 13.

(c)



(d)

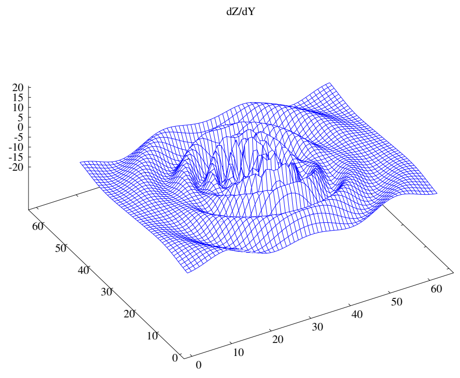
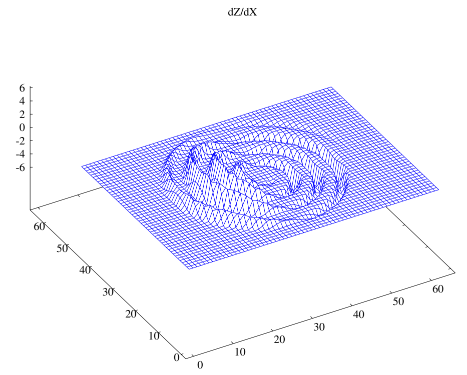


Figure 20: Analytic slope data for height function 08 (concentric ripples): X derivative f (a) and Y derivative g (b). Compare with figure 15.

(c) (d)



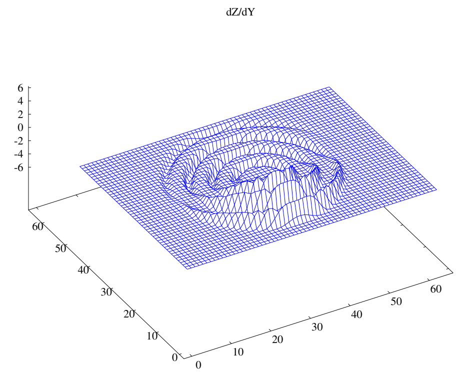
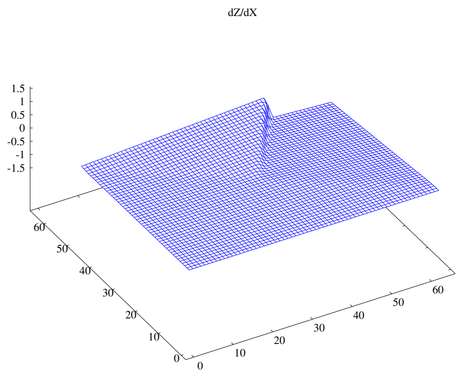


Figure 21: Analytic slope data for height function 09 (Babel tower): X derivative f (a) and Y derivative g (b). Compare with figure 16.

(c) (d)



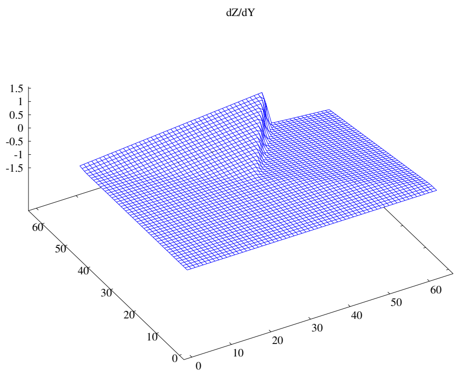


Figure 22: Analytic slope data for height function 10 (divergent parabolic ramps): X derivative f (a) and Y derivative g (b). Compare with figure 17.

4.2.3 Slope reliability weights

For tests with weighted data, the slope reliability weight $w[u, v]$ of each pixel was determined by comparing the analytic gradient (computed as in section 4.2.2) in the pixel with the corresonding numerical gradient (computed from the height map as in section 4.2.1). Specifically, the weight was set to $\max\{0, (1 - \delta/\delta_0)^2\}$, where δ was the Euclidean distance between the two gradients, and δ_0 a threshold (1.0 in the case of function 10).

4.3 Test results with synthetic data

The outputs of our algorithm on the previously described datasets are shown in sections 4.3.1 through 4.3.5. In each figure, plot (a) shows the height map $z[u, v]$ computed by our program, and plot (b) shows the height error map — namely the the difference between the computed heights $z[u, v]$ and the corresponding reference heights $z^*[u, v]$, after both were adjusted to have zero mean.

4.3.1 Tests with numerical slopes, unweighted

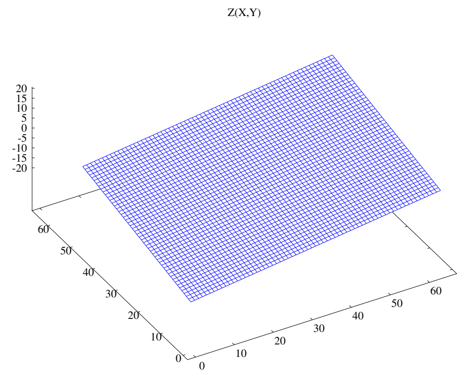
In the first batch of tests we used the numerical slopes derived from the reference height field, as explained in section 4.2.1. All reliability weights $w[u, v]$ set to 1. The results are shown in figures 23 through 30.

Observe that the errors are generally much smaller than one height unit (for a total range of about 10 to 20 height units), and are largely confined to the places where the slope changes abruptly. Observe also that the height maps derived from the linear ramp function (figure 23) were reconstructed exactly, showing that the multiscale integration correctly recovers the linear term that is lost in the Laplacian formulation.

The largest errors occurred in the reconstruction of function 08 (concentric ripples), as shown in figure 28. In that test, the spacing of the slope samples was too coarse to adequately capture the ripples near the center of the height map, where the wavelength is only a couple of pixels. Nevertheless, observe that those inconsistencies did not affect the integration in the surrounding areas, where the ripples could be sampled more accurately.

Note that the method accurately reconstructed function 10, in spite of its sharp discontinuity. This is due to the fact that the numerical slopes take that discontinuity into account.

(e)



(f)

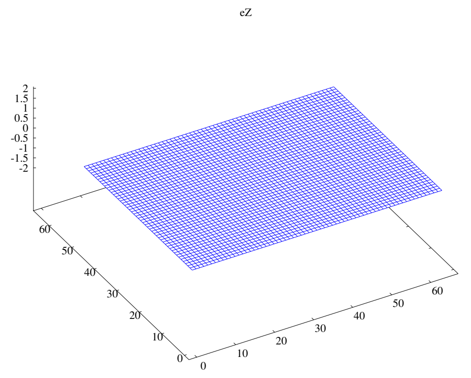
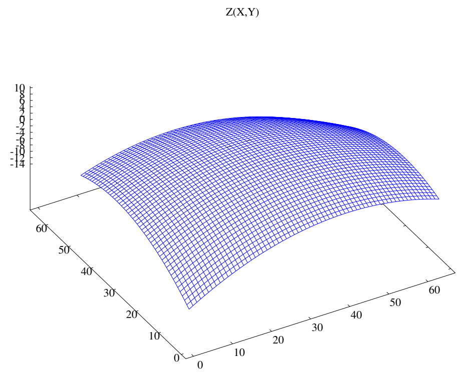


Figure 23: Output of test for height function 03 (linear ramp), from numerical slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 2.

(e)



(f)

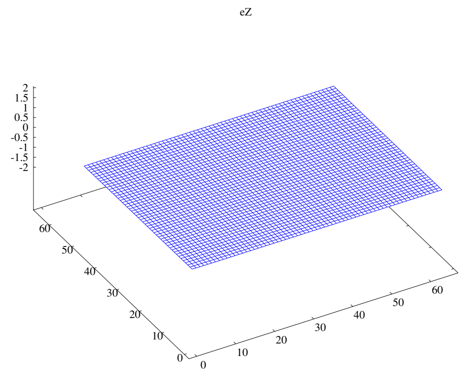
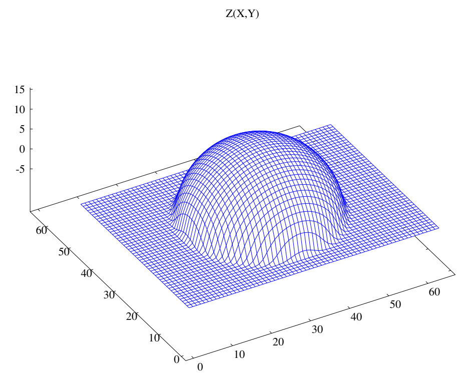


Figure 24: Output of test for height function 04 (quadratic hump), from numerical slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 3.

(e)



(f)

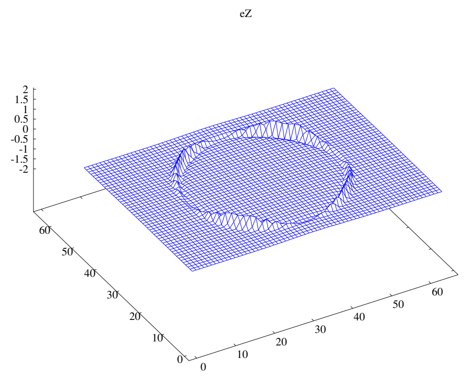
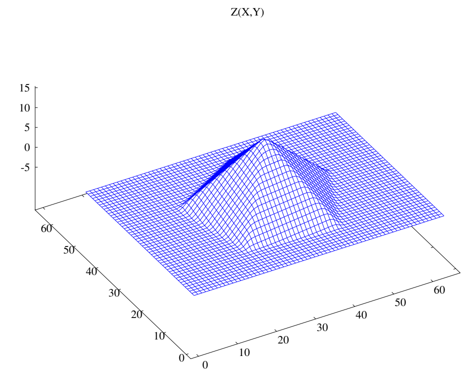


Figure 25: Output of test for height function 05 (hemispherical dome), from numerical slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 4.

(e)



(f)

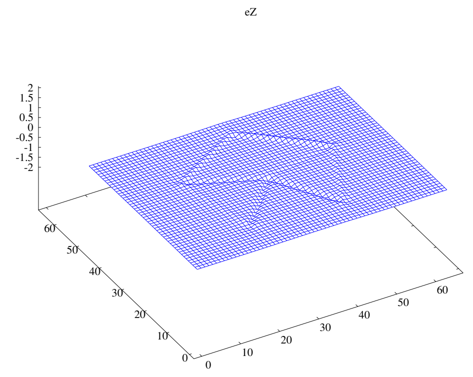
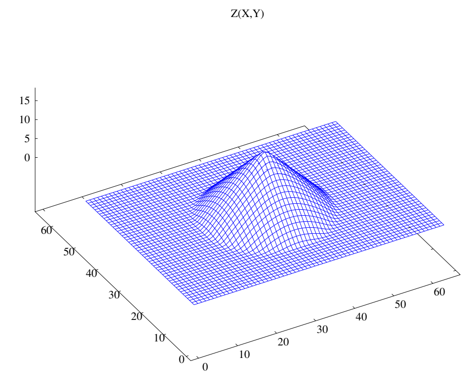


Figure 26: Output of test for height function 06 (five-sided pyramid), from numerical slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 5.

(e)



(f)

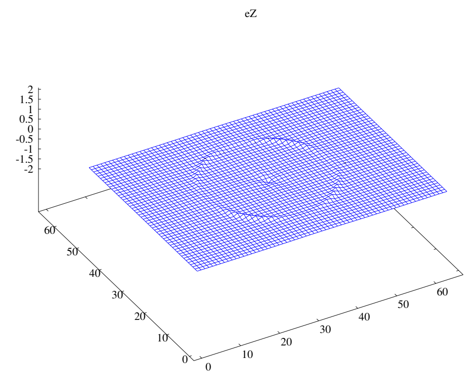
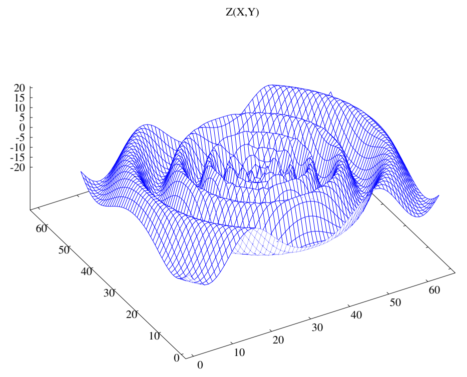


Figure 27: Output of test for height function 07 (conical mound), from numerical slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 6.

(e)



(f)

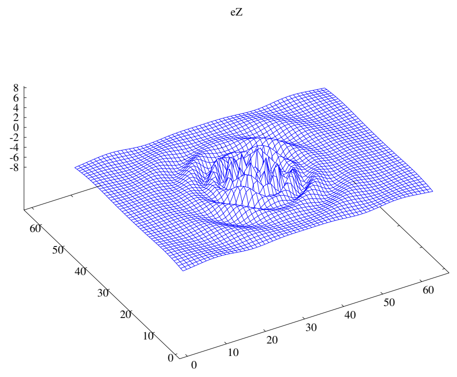
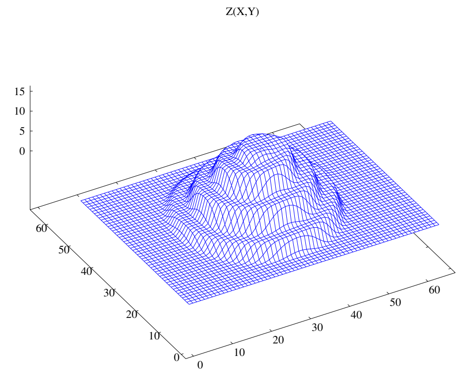


Figure 28: Output of test for height function 08 (concentric ripples), from numerical slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 7.

(e)



(f)

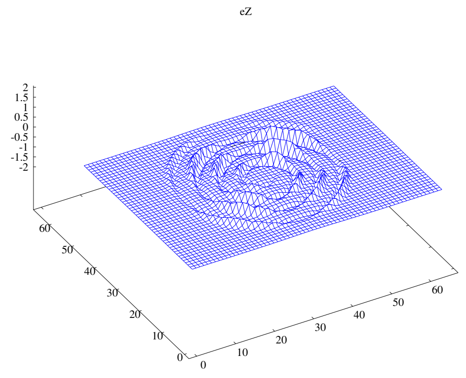
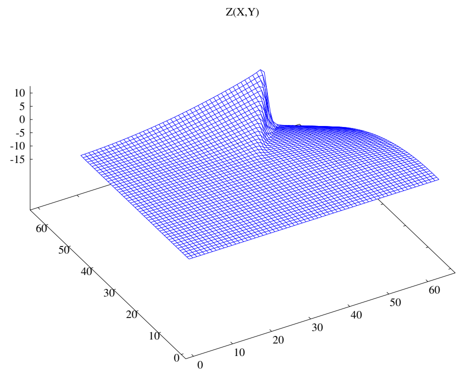


Figure 29: Output of test for height function 09 (Babel tower), from numerical slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 8.

(e)



(f)

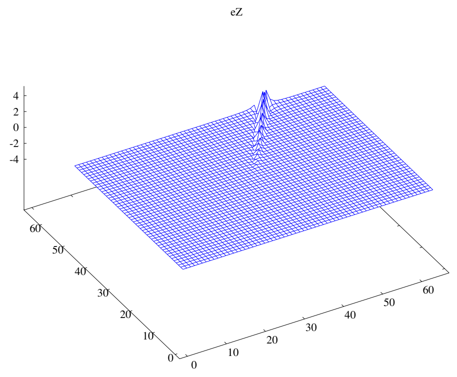


Figure 30: Output of test for height function 10 (divergent parabolic ramps), from numerical slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 9.

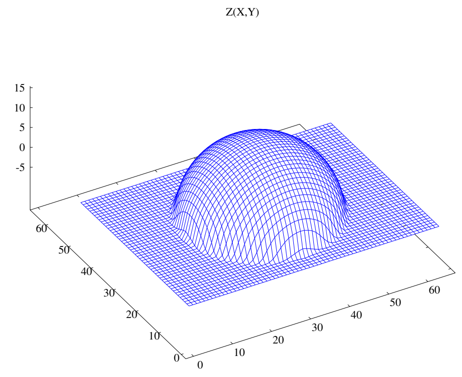
4.3.2 Tests with analytic slopes, unweighted

In the next batch of tests, we used the analytic slopes, as defined in section 4.2.2. For testing purposes we set all reliability weights $w[u, v]$ to 1, even though the analytic slopes were meaningless along zero-order discontinuities (such as of function 10, and somewhat unreliable along first-order discontinuities (such as the perimeter of the hemispherical dome in figure 4) or in regions where the height varied very rapidly (such as near the center in figure 7).

Some of the results are shown in figures 31 through 35. (The results for functions 03, 04 and 07 are very similar to those in the previous batch, and were omitted to save space.)

Except for function 10, all height maps were reconstructed with about the same accuracy as when using numeric slopes. In the case of function 10, note in figure 35 that the misleading slope data around the discontinuity line caused the integrator to produce a very incorrect, almost flat height field.

(e)



(f)

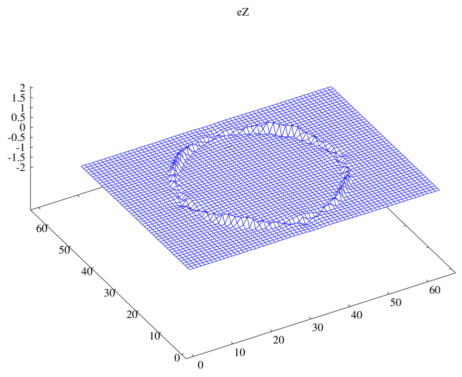
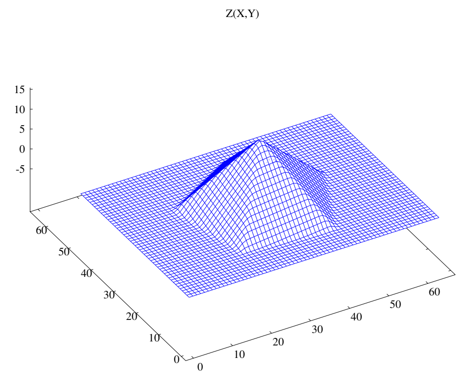


Figure 31: Output of test for height function 05 (hemispherical dome), from analytic slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 25.

(e)



(f)

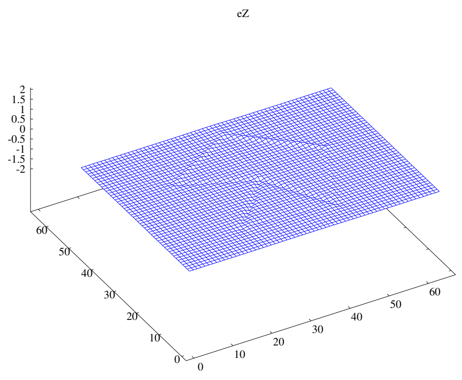
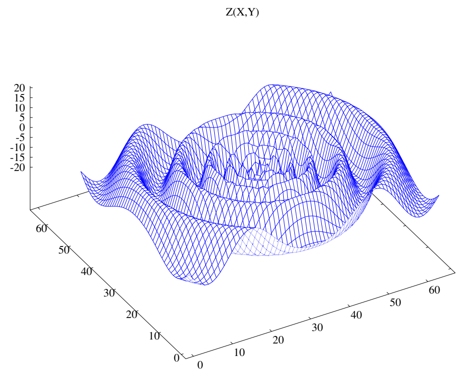


Figure 32: Output of test for height function 06 (five-sided pyramid), from analytic slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 26.

(e)



(f)

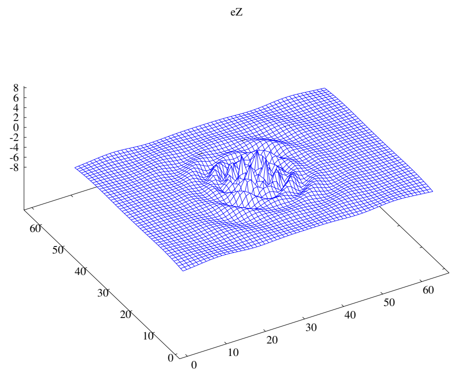
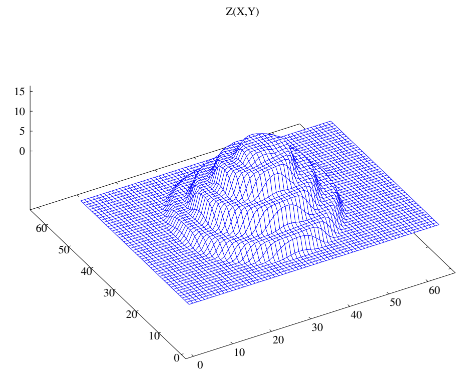


Figure 33: Output of test for height function 08 (concentric ripples), from analytic slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 28.

(e)



(f)

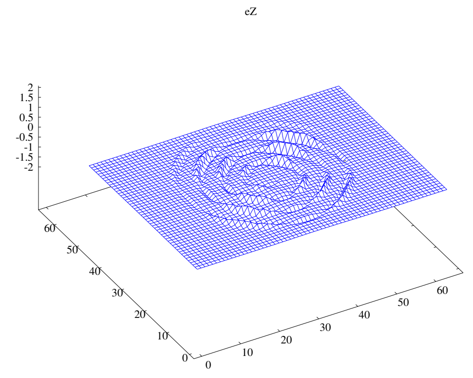
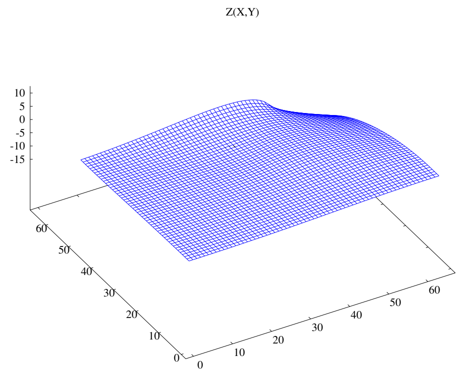


Figure 34: Output of test for height function 09 (Babel tower), from analytic slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 29.

(e)



(f)

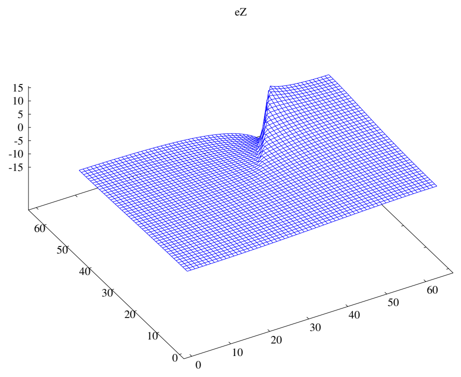


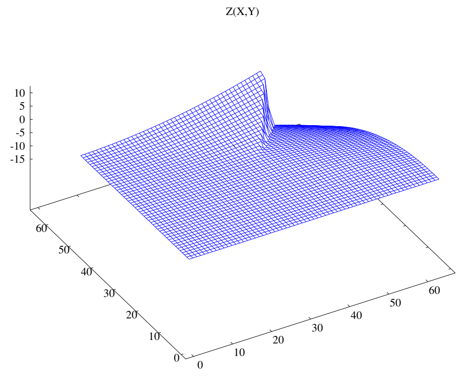
Figure 35: Output of test for height function 10 (divergent parabolic ramps), from analytic slopes with uniform weights: computed height map (a), height error map (b). Compare with figure 30.

4.3.3 Tests with analytic slopes, weighted

The following batch of tests illustrate the use of the reliability weights in the handling of discontinuities, specifically that of function 10 (divergent ramps).

Figure 36 shows the result of integrating the analytic slope maps of function 10 with these reliability weights. By marking the slope data along the cliff as unreliable, the final heights were determined by integration of the slopes along paths that avoided that discontiuity. Figure 36 should be compared to figure 35, which uses exactly the same slope data but uniform weights.

(e)



(f)

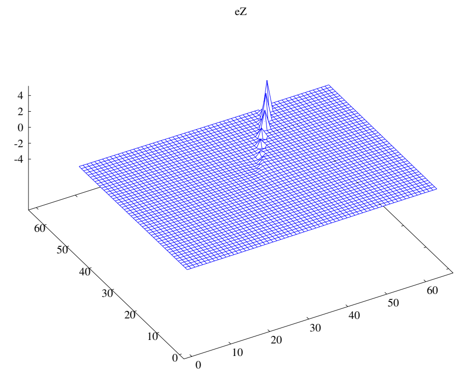
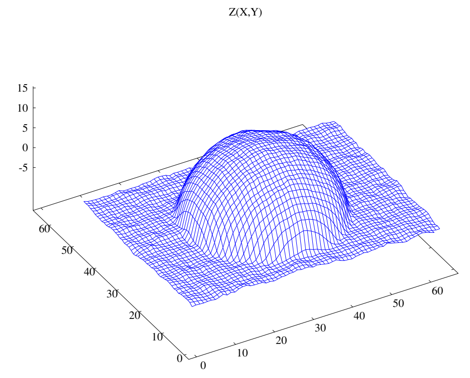


Figure 36: Output of test for height function 10 (divergent ramps), from analytic slopes with proper reliability weights: computed height map (a), height error map (b). Compare with figure 35.

4.3.4 Tests with noisy data

To evaluate the robustness of the algorithm, we also applied it to noisy slope maps. Specifically, we added to each derivative datum $f[u,v]$ or $g[u,v]$ a random real value drawn from a Gaussian distribution with mean zero and standard deviation $\sigma = 0.30$. A typical result is shown in figure 37. Note that the error in the height field remains below 0.6 height unit. Observe that equation (3) was generally not satisfied in this test, since the perturbations added to $f[u,v]$ and $g[u,v]$ were independent of each other.

(e)



(f)

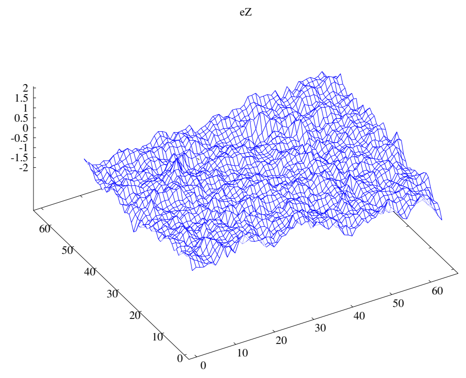
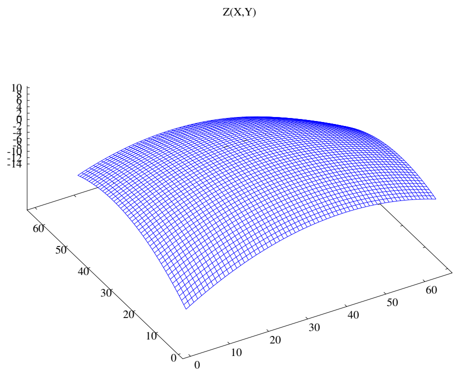


Figure 37: Output of test for height function 05 (hemispherical dome), from analytic slopes perturbed by Gaussian noise ($\sigma = 0.3$) and proper reliability weights: computed height map (a), height error map (b). Compare with figure 37.

4.3.5 Tests with noisy weights

Figure 38 shows the outcome of another test where the slope data f and g were noise-free, but the weights were randomly chosen so that their logarithms had a Gaussian distribution with mean 0 and deviation $\sigma = 1.00$. This result should be compared to figure 24. Note that the perturbation of the weights had practically no effect in the result.

(e)



(f)

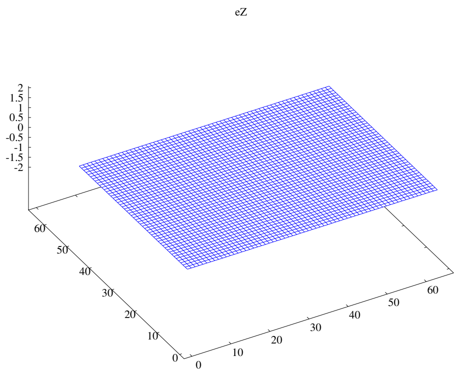


Figure 38: Output of test for height function 04 (quadratic hump), from analytic slopes with log-Gaussian random weights ($\sigma = 1.00$): computed height map (a), height error map (b). Compare with figure 24.

4.4 Tests with real object

Figure 39 shows the outcome of another test where the slope data f and g were obtained from a real object by phtometric stereo methods. The object, whose perspective view is shown in figure 39(a), was a plaster sculpture stained with non-glossy tempera paint of various colors. The slope maps were obtained by analysis of frontal photos of the object under various lightings, as described by Leitão, Saracchini and Stolfi [1].

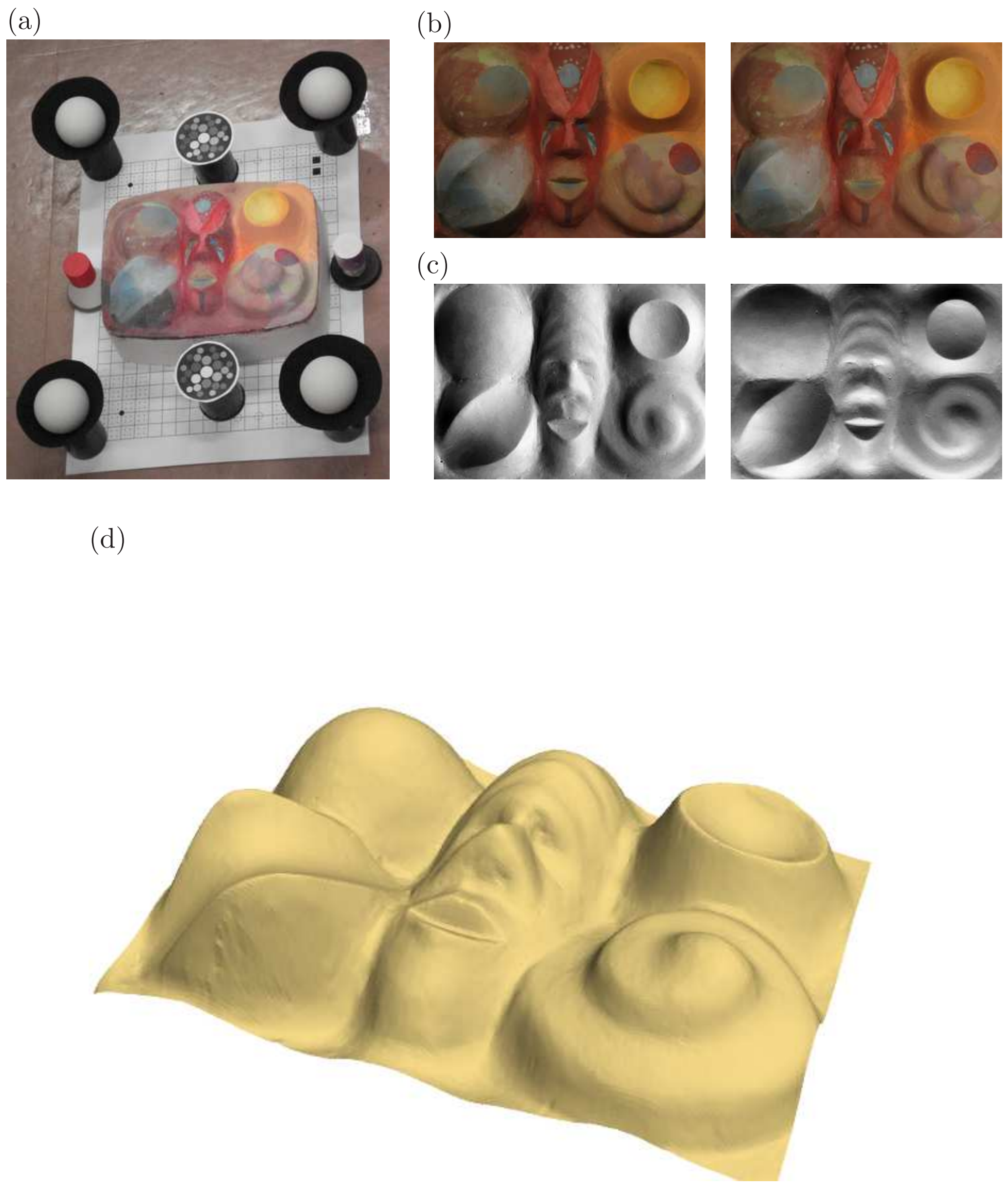


Figure 39: Test with real object: perspective photo of object surrounded by calibration objects and exposure charts (a), two of the 24 frontal images (b), input X and Y slope maps obtained by photometric stereo from the frontal images (c), and the height map computed by our algorithm, rendered with POV-Ray (d).

5 Computing slopes from normals

Photometric stereo methods sometimes yield the average normal vector $\vec{n}[u, v]$ around each point $p[u, v]$, instead of the average x and y slopes. In those situations we must convert the normals $\vec{n}[x, y]$ to the slopes f_{++} and g_{++} before feeding them to our integration algorithm. (Note that the integration operator is linear when the input are the slopes, but non-linear when the inputs are the surface normals. Moreover, when reducing the problem to a smaller scale, the slopes within a 2×2 block can be directly averaged, whereas the normals cannot.)

For simplicity we assume that the camera is at infinite distance from the subject, on the z axis. The slopes are computed as

$$f_{++} \leftarrow -\frac{\vec{n}[u,v].\mathrm{x}}{\vec{n}[u,v].\mathrm{z}} \qquad g_{++} \leftarrow -\frac{\vec{n}[u,v].\mathrm{y}}{\vec{n}[u,v].\mathrm{z}} \tag{25}$$

Since the surface normal $\vec{n}[u,v]$ is averaged over the *visible* points of the scene around $p[u,v]$, it cannot be perpendicular to the viewing direction (the z axis) or point away from it. Therefore $\vec{n}[u,v].\mathrm{z}$ must be positive. If $\vec{n}[u,v].\mathrm{z}$ is zero or negative, the slopes f_{++} and g_{++} may be set to any arbitrary value (such as 0) and the weight w_{++} must be set to 0.

6 The program

Our implementation of the algorithm is available as a set of libraries and a main program (`slope_to_height`) written in the GNU dialect of C. The code is available at the URL `http://www.ic.unicamp.br/~stolfi/EXPORT/projects/photo-stereo/`. The program reads an array with $2 \times m \times n$ `float` values, containing the slope data $f[u,v]$ and $g[u,v]$ at each slope sampling point $p[u,v]$. The program optionally reads a separate array containing the slope reliability weights $w[u,v]$ at those same points. The companion programs `fni_to_pnm` and `pnm_to_fni` allow conversion of these array files to and from the popular PBMplus image format [2].

7 Conclusions

We have described in this report an algorithm that integrates a slope (gradient) map to yield a height (depth) map. Thanks to the use of multiscale integration, our method is exceedingly fast, robust and accurate. Moreover, by accepting reliability weights for individual slope data, our method allows reliable reconstruction of discontinuous height fields.

References

[1] Helena Cristina da Gama Leitão, Rafael F. V. Saracchini, and Jorge Stolfi. A bucket grid structure to speed up table lookup in gauge-based photometric stereo. In *Proceedings of the 20th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2007)*, pages 221–227. IEEE Computer Society Press, October 2007.

[2] Bryan Henderson. Netpbm home page. Electronic document at `http://netpbm.sourceforge.net/`, edit date 2008/08/01, last accessed on 2008/10/22, August 2008.

[3] Berthold K. P. Horn and Michael J. Brooks. *Shape from Shading.* MIT Press, Cambridge, Mass., 1989.

[4] Berthold K. P. Horn, Richard J. Woodham, and William M. Silver. Determining shape and reflectance using multiple images. Technical Report AI Memo 490, MIT Artificial Intelligence Laboratory, 1978.

[5] Reinhard Klette and Karsten Schlüns. Height data from gradient fields, 1996.

[6] Bertold K. P.Horn. Height and gradient from shading. Technical Report AI Memo 1105, Massachusetts Institute of Technology, 1989.

[7] Antonio Robles-Kelly and Edwin R. Hancock. Surface height recovery from surface normals using manifold embedding. In *Proceedings of the 2004 International Conference on Image Processing (ICIP)*. IEEE, 2004.

[8] Gavin D.J. Smith and Adrian G. Bors. Height estimation from vector fields of surface normals. In *Proc. 2002 IEEE International Conference on Digital Signal Processing (DSP'02)*, pages 1031–1034, 2002.

[9] D. Terzopoulos. The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):417–438, 1988.

[10] Tiangong Wei and Reinhard Klette. Height from gradient using surface curvature and area constraints. In *Proc. 3rd Indian Conference on Computer Vision, Graphics and Image Processing*, 2002.

[11] Robert J. Woodham. Photometric method for determining suface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.