INSTITUT FÜR INFORMATIK

UND PRAKTISCHE MATHEMATIK

Geometrical Enforcement of Integrability and 2-D Leap-Frog

Lyle Noakes and Ryszard Kozera

Bericht Nr. 2006
May 2000

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

KIEL

Institut für Informatik und Praktische Mathematik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Geometrical Enforcement of Integrability and 2-D Leap-Frog

Lyle Noakes and Ryszard Kozera

Bericht Nr. 2006
May 2000

e-mail: {lyle@maths.}{ryszard@cs.}.uwa.edu.au

This work has been supported by Alexander von Humboldt Foundation.

Geometrical Enforcement of Integrability and 2-D Leap-Frog

Lyle Noakes[1*] Ryszard Kozera[2†]

[1]Department of Mathematics and Statistics
The University of Western Australia
Nedlands, WA 6907
Australia
e-mail: lyle@maths.uwa.edu.au
fax: +61 89 380 1028

[2]Department of Computer Science
The University of Western Australia
Nedlands, WA 6907
Australia
e-mail: ryszard@cs.uwa.edu.au
fax: +61 89 380 1089

Abstract

The 1-D Leap-Frog Algorithm of Noakes [1] is an iterative scheme for solving particular nonlinear optimization problems. We aim to adapt 1-D Leap-Frog to optimization problems in computer vision and the present paper represents a step in that direction. In the context of photometric stereo we use a 2-D version of Leap-Frog to compute optimal integrable approximations of noisy nonintegrable vector fields. 2-D Leap-Frog is iterative and converges linearly to the optimal field. As evident in [1], Leap-Frog can sometimes deal with nonlinear problems, and this feature might in future be important for computer vision. However, even the capacity to handle large systems of equations is important in vision applications, and the present paper demonstrates Leap-Frog's capacity to do that, at least when the equations are linear. In this situation Leap-Frog can be viewed as an extension of Gauss-Seidel. Properties of these kinds of extensions are well-known but we offer a particularly simple geometrical proof for the special case of photometric stereo. From the point of view of vision the present paper marks a significant advance over known methods in photometric stereo of Noakes, Kozera and Klette [2], Frankot and Chellappa [3], and of Horn [4]. The performance of 2-D Leap-Frog was demonstrated in Noakes and Kozera [5] without proof of convergence.

Keywords: computer vision, integrability, global optimization, vector fields, surface reconstruction and photometric stereo.

*Research supported by The Australian Research Council.

†Research supported by The Australian Research Council and The Alexander von Humboldt Foundation, Bonn.

1 Introduction

A monochrome picture of a smooth object typically exhibits brightness variation or *shading*. *Shape-from-shading* is the problem of determining the visible part of the object from the picture. As shown by Horn [6] subsection 10.10 (see also Horn and Brooks [7]) this corresponds to solving a nonlinear first-order partial differential equation. Specifically, one seeks a function u, representing surface depth in the direction of the $z-$axis, satisfying a so-called *image irradiance equation*. In the case of *a Lambertian surface* (a perfect light diffuser), illuminated from the light-source direction (p_1, p_2, p_3), the image irradiance equation is

$$\frac{p_1 u_x(x,y) + p_2 u_y(x,y) - p_3}{\sqrt{p_1^2 + p_2^2 + p_3^2}\sqrt{u_x^2(x,y) + u_y^2(x,y) + 1}} = E(x,y) \tag{1}$$

over an image $\Omega \subset \mathbb{R}^2$. Here E is an image intensity formed by orthographic projection onto the plane of the image Ω. The unknown surface S is the graph of the function u, and is to be determined up to translations $u \mapsto u + c$. A single image shape-from-shading problem (with no additional constraints) is, in general *ill-posed* in the sense that not enough data is given for the problem (1) to be uniquely solved. For background we refer to Brooks and Chojnacki [8], Brooks, Chojnacki, and Kozera [9–11], Deift and Sylvester [12], Dupuis and Oliensis [13], Kimmel and Bruckstein [14], Klette, Schlüns, and Koschan [15], Kozera [16–19], Oliensis [20], Onn and Bruckstein [21] and Rouy and Tourin [22].

In contrast with single image shape from shading, in *two-source photometric stereo* the shape of a Lambertian surface is generically uniquely determined by a pair of images obtained by consecutive illuminations of a given scene from two distinct light-source directions (see Horn [6, subsection 10.16], Kozera [16,17] and Onn and Bruckstein [21]). When 3 light-source directions are used we have *three-source photometric stereo* and the additional information has many uses. For the present paper it is enough to consider two-source photometric stereo.

Two-source photometric stereo for a Lambertian surface, is modelled by two nonlinear PDEs of the form (1), where the right hand sides E_1, E_2 are given functions defined on $\Omega = \Omega_1 \cap \Omega_2 \subset \mathbb{R}^2$. Shape reconstruction, as in mutiple-source photometric stereo, can be decomposed into

- *gradient computation* (an algebraic step) and
- *gradient integration* (an analytic step).

It turns out (see subsection 10.16 in Horn [6], Kozera [16, 17] and Onn and Bruckstein [21]) that the gradient $\vec{v}_{grad} = (u_x, u_y)$ can be generically uniquely computed from the data E_1, E_2 and the image irradiance equations. What complicates our problem, especially in the presence of noise, is that the tableaux of computed values v^1, v^2 of u_x, u_y usually does not correspond to a function u. To see why not we suppose that u is C^2. Then

$$u_{xy}(x,y) = u_{yx}(x,y).$$

It follows that a necessary condition for the v^1, v^2 to correspond to a C^2 function is

$$v^1_y = v^2_x. \tag{2}$$

When Ω is simply-connected this *integrability condition* is also sufficient. From now on take Ω to be the unit square $[0,1]\times[0,1]$. There are integrability conditions for C^1 functions, and also for nonsimply-connected domains, but (2) is enough for us. As noted, the integrability condition usually does not hold for computed v^1, v^2 in the presence of noisy measurements E_1, E_2. If possible this defect should be corrected before passing to the second step in surface reconstruction, namely gradient integration. Then gradient integration becomes extremely simple, and u is determined up to translation. Namely, once the gradient (u_x, u_y) is computed we obtain u according to the formula

$$u(x,y) = u(x_0,y_0) + \int_\gamma u_x dx + u_y dy, \tag{3}$$

where $\gamma \subset \Omega$ is an arbitrary piece-wise C^1 curve joining a fixed point $(x_0, y_0) \in \Omega$ with $(x,y) \in \Omega$. The choice of (x_0, y_0) determines the translation constant c.

The essential remaining problem then is how to replace the vector field (v^1, v^2) by an integrable field $(\hat{v}^1, \hat{v}^2)$. Perhaps the most obvious way to do this is to replace (v^1, v^2) by the closest integrable vector field, and of course it is important to say what is meant by *close* in this context. In the present paper we use the standard definition, which is convenient from the mathematical point of view [1]. Techniques developed so far for handling this problem include Horn [4, 6] subsections 11.7-11.8 which uses a method that relies on minimizing different best fit functionals. For example, one of Horn's variants minimizes the following functional

$$\int_\Omega ((u_x(x,y) - v^1(x,y))^2 + (u_y(x,y) - v^2(x,y))^2) d\Omega$$

whose Euler-Lagrange equation is a Poisson equation $\triangle u = f$. The problem with this approach is the need for additional boundary conditions, which do not naturally occur for the problem as stated. The choice of boundary conditions affects the solution and so in practice there are some problems with this method. Another minimization scheme was introduced by Frankot and Chellappa [3], where the projection to the closest function $\tilde{u}$ expanded by Fourier series is used. Such methods work best when $\tilde{u}$ is periodic, and we wish to avoid this kind of constraint. Recently a new method for enforcing integrability of noisy vector fields, namely the *Lawn-Mowing Algorithm* of Noakes, Kozera, and Klette [2] has been introduced. Like the other methods considered so far, Lawn-Mowing is serviceable but suboptimal.

The 2-D Leap-Frog used in the present paper suffers from none of the difficulties mentioned above, and moreover we prove linear convergence to the optimal

[1] In subsequent papers we shall study alternative definitions of distance, more closely tied to applications, and requiring the power of Leap-Frog to handle nonlinearities.

solution. Our geometrical proof seems more accessible than well-known algebraic proofs of convergence for classical iterative schemes solving large systems. 2-D Leap-Frog (at least in the present linear setting) can be viewed as a subclass of such methods. As already noted, 2-D Leap-Frog is derived from 1-D Leap-Frog which is an iterative scheme for solving nonlinear optimization problems in a geometrical setting. This is why our proofs are geometrical in flavour (even in the linear case) and why we hope this style of algorithm will be useful for more realistic (nonlinear) optimization problems arising in computer vision. As well as proof of convergence we give a number of illustrative examples, pictures and pertinent experiments.

2 Pseudoinverses

We first recall some linear algebra. For $n > m$ let $L : \mathbb{R}^n \to \mathbb{R}^m$ be a linear transformation of maximal rank m. Given $L(\vec{v}) \neq \vec{u}$, for some $\vec{v} \in \mathbb{R}^n$ and $\vec{u} \in \mathbb{R}^m$, consider the task of finding the closest $\hat{\vec{v}} \in \mathbb{R}^n$ to $\vec{v} \in \mathbb{R}^n$ such that $L(\hat{\vec{v}}) = \vec{u}$. It is well-known that $\|\vec{v} - \hat{\vec{v}}\|$ is minimized by the orthogonal projection of $\vec{v}$ on the affine subspace

$$A_L = \vec{v}_p + Ker(L) \subset \mathbb{R}^n, \tag{4}$$

where $Ker(L)$ is the kernel of L and $\vec{v}_p$ is a particular solution to the problem $L(\vec{x}) = \vec{u}$.

In order to actually find $\hat{\vec{v}}$ we need to compute its coordinates $\hat{\alpha}_i$ with respect to some given basis $\{\vec{f}_1, \vec{f}_2, \ldots \vec{f}_k\}$ of $Ker(L)$, where $k = n - m$. Then the $\hat{\alpha}_i$ are solutions of the following $k \times k$-linear system:

$$\sum_{i=1}^{k} \hat{\alpha}_i < \vec{f}_i | \vec{f}_j > = < \vec{v} - \vec{v}_s | \vec{f}_j >,$$

in principle solvable by Gaussian elimination. When k is large and the matrix of L has special properties, Gauss-Seidel or, more generally, multiplicative Schwarz (see Benzi, Nabben and Szyld [23]) may be more appropriate. An alternative approach is to use the *pseudoinverse*. For this the following lemma is used (the straightforward proof is omitted).

Lemma 1 *The vector $\vec{v} - \hat{\vec{v}}$ is orthogonal to $Ker(L)$ if and only if there exists $\vec{w} \in \mathbb{R}^m$ such that $\vec{v} - \hat{\vec{v}} = L^T(\vec{w})$.*

Since $rank(L) = m$, the symmetric matrix $LL^T : \mathbb{R}^m \to \mathbb{R}^m$ is positive-definite and in particular is nonsingular. Consequently, since $(LL^T)^{-1}$ is well-defined and as $L(\hat{\vec{v}}) = \vec{u}$, there is a unique vector $\vec{w}$ such that $\vec{v} - \hat{\vec{v}} = L^T(\vec{w})$. It can be easily shown that vector $\vec{w}$ satisfies

$$\vec{w} = (LL^T)^{-1}(L(\vec{v}) - \vec{u}).$$

So the corrected vector $\hat{\vec{v}}$ can be expressed as

$$\hat{\vec{v}} = \vec{v} - L^T(LL^T)^{-1}(L(\vec{v}) - \vec{u}). \tag{5}$$

The transformation $L_{ps} = L^T(LL^T)^{-1} : \mathbb{R}^m \to \mathbb{R}^n$ is the *pseudoinverse* of L. As with ordinary inverses it is not always convenient to use the pseudoinverse. Direct or iterative methods of solution may be more efficient. From this point of view, 2-D Leap-Frog is an iterative scheme, particularly well suited to photometric stereo and with links to other problems of interest.

3 The 2-D Leap-Frog Global Optimizer

In practice, our problem of approximating a noisy vector field by an integrable field can be effectively reduced to the linear algebra task in section 2, by transforming the integrability condition (2) into its discrete analogue. In more detail, for fixed $k < l \in \mathbb{N}$ divide the unit square Ω into 2^{2l} *atomic subsquares* of the form $S^l_{i_q j_g} = [(i_g - 1)/2^l, i_g/2^l] \times [(j_g - 1)/2^l, j_g/2^l]$ (for $1 \leq i_g, j_g \leq 2^l$). From values of u on such a grid we could calculate central-difference derivative approximations with $\Delta x = \Delta y = 1/2^l$

$$u_x[i,j] \approx \frac{u^j_{i+1} - u^j_i}{2^l} \quad \text{and} \quad u_y[i,j] \approx \frac{u^{j+1}_i - u^j_i}{2^l}$$

for each side of a subsquare in the x and y directions, accordingly whether the side is horizontal or vertical. Along each atomic subsquare, the analytic integrability condition (2) (modulo truncation error assumed here to be dominated by noise) translates into the *discrete analogue*

$$u_x[i, j+1] - u_x[i,j] = u_y[i+1, j] - u_y[i,j]. \tag{6}$$

Assume that uniform Gaussian noise with mean zero is added to each $u_x[i,j]$ and each $u_y[i,j]$ independently. The problem now is to estimate u_x, u_y from $2^{l+1}(2^l+1)$ noise contaminated differences in such a way as to be closest to the nonintegrable field $\vec{v} = (v^1, v^2)$ evaluated at the gridpoints. Accordingly, $\hat{\vec{v}} = (\hat{u}_x, \hat{u}_y)$ should minimize the sum of squared residuals

$$\sum_{1 \leq i,j \leq 2^l(2^l+1)} \left((\hat{u}_x[i,j] - v^1[i,j])^2 + (\hat{u}_y[i,j] - v^2[i,j])^2 \right). \tag{7}$$

Such a solution is said to be *optimal*. Finding it is a task in linear algebra, as outlined in section 2 and, as noted, this leads to a large system of linear equations in many unknowns, requiring about 70GB of RAM for typical problems if implemented directly. The need for some sort of iterative scheme is quite apparent.

There is a well-known relationship between minimization of L^2-norm and maximum likelihood estimates of parameters from measurements contaminated by

Gaussian noise (see Zubrzycki [24]§51). So the (standard) assumption about the Gaussian noise tends to simplify the analysis.

In the next subsection we confine attention to a given S_{ij}^{kl} (denoted also by S_t) having 2^{2k} atomic squares, and review various optimization problems considered over S_t subject to nine different boundary conditions, namely: *top-right, left-top-right, left-top, top-right-bottom, top-right-bottom-left, bottom-left-top, right-bottom, right-bottom-left, and bottom-left.* These are referred to in the description of 2-D Leap-Frog in section 3.3.

3.1 Subsquare Grids with Various Boundary Constraints

We briefly discuss one of nine cases of S_t-boundary constraints. In doing so, we assign to the nonintegrable vector field $\vec{v}_h = (v^1[i,j], v^2[i,j])$ (over each S_t) $2^{k+1}(2^k+1)$ free variables $\vec{v}_z$ corresponding to the unknown corrected values of the closest integrable vector field $\hat{\vec{v}}_h$. More specifically, over each S_t, define $2^k(2^k+1)$ free variables (corresponding to $v^1[i,j]$), namely: $x_1 = v^1[0,0], \ldots, x_{2^k} = v^1[2^k - 1, 0], x_{2^k+1} = v^1[0,1], \ldots, x_{2^{k+1}} = v^1[2^k-1, 1]$, and $x_{2^{2k}+1} = v^1[0, 2^k], \ldots, x_{2^k(2^k+1)} = v^1[2^k-1, 2^k]$. Analogously, we have $2^k(2^k+1)$ variables (corresponding to $v^2[i,j]$), namely: $y_1 = v^2[0,0], \ldots, y_{2^k} = v^2[0, 2^k-1], y_{2^k+1} = v^2[1,0], \ldots, y_{2^{k+1}} = v^2[1, 2^k - 1], y_{2^{2k}+1} = v^2[2^k, 0], \ldots, y_{2^k(2^k+1)} = v^2[2^k, 2^k - 1]$. Assuming temporarily, that $k = l$ the integrability condition (6) applied to each atomic subsquare yields a homogeneous optimization system of 2^{2l} linear equations in $2^{l+1}(2^l+1)$ unknowns

$$L_l^h(\vec{v}_z) = \vec{0},$$

where $\vec{0} \in \mathbb{R}^{2l}$ and $\vec{v}_z \in \mathbb{R}^{2^{l+1}(2^l+1)}$ with $L_l^h(\vec{v}_h) \neq \vec{0}$ (note that for $l = k$ we have $\vec{v}^p = \vec{0}$ as defined in (4)). Direct methods for solving such a global optimization problem (for which $L_l^h(\hat{\vec{v}}_h) = \vec{0}$) constitute, for $l \geq 7$, an unwieldly computational task for currently available computers ($l = 7$ or 8 or 9 corresponds to a typical camera resolution). So further discussion focuses on local optimizations over each S_t (with $k < l$) and on melding local optima into a global optimum (the 2-D Leap-Frog).

As the analysis of nine cases of different boundary constraints imposed on S_t for the 2-D Leap-Frog is similar we discuss here only the case of *top-right* boundary conditions. Assume that 2^{k+1} boundary values are given, *i.e.* $x_{2^{2k}+1} = x^0_{2^{2k}+1}, x_{2^{2k}+2} = x^0_{2^{2k}+2}, \ldots, x_{2^k(2^k+1)} = x^0_{2^k(2^k+1)}$ (representing top boundary conditions for S_t) and $y_{2^{2k}+1} = y^0_{2^{2k}+1}$, $y_{2^{2k}+2} = y^0_{2^{2k}+2}$, $\ldots, y_{2^k(2^k+1)} = y^0_{2^k(2^k+1)}$ (representing right boundary conditions for S_t). Applying 2^{2k} integrability constraints (6) (along each atomic square) we arrive at an inhomogeneous system of 2^{2k} linear equations in $2^{k+1}(2^k+1) - 2^{k+1} = 2^{2k+1}$ unknowns. Using notation from section 2 this system can be treated as $L_k^{tr}(\vec{x}_{tr}) = \vec{u}_{tr}$ (for $L_k^{tr}(\vec{v}_{tr}) \neq \vec{u}_{tr}$), with $\vec{x}_{tr}, \vec{v}_{tr} \in \mathbb{R}^{2^{2k+1}}$, $\vec{u}_{tr} \in \mathbb{R}^{2^{2k}}$ and $L_k^{tr} : \mathbb{R}^{2^{2k+1}} \to \mathbb{R}^{2^{2k}}$ being a linear operator. Note that for $n = 2^{2k+1}$ and $m = 2^{2k}$ condition $n > m$ holds. It is straightforward to show that $Rank(L_k^{tr}) = m$. Consequently, as $L_k^{tr}(\vec{v}_{tr}) \neq \vec{u}_{tr}$, the closest vector $\hat{\vec{v}}_{tr}$ satisfying $L_k^{tr}(\vec{x}_{tr}) = \vec{u}_{tr}$ can be found as in section 2. Note that as k increases

the dimensions of both linear spaces $\mathbb{R}^m$ and $\mathbb{R}^n$ grow exponentially. The original problem (7) is reduced to a collection of computationally tractable problems, where l is replaced by $1 \leq k < l$. In the case of $k = 1$ we have $L_1^{tr}(\vec{x}_{tr}) = \vec{u}_{tr}$, where $L_1^{tr} : \mathbb{R}^8 \to \mathbb{R}^4$ is defined as

$$L_1^{tr} = \begin{pmatrix} 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix},$$

$\vec{x}_{tr} = (x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4)$, and $\vec{u}_{tr} = (0, -y_5^0, x_5^0, x_6^0 - y_6^0) \in \mathbb{R}^4$. Analogously, for any $1 < k < l$ one can find an explicit formula for L_k^{tr}. For example, for $k = 2$, an explicit formula determining $L_2^{tr} : \mathbb{R}^{32} \to \mathbb{R}^{16}$ can still be quickly established by hand. It is evident, however, that finding the matrix L_k^{tr} (for $k > 2$) consitutes a more laborious task. In a similar manner the operators $L_k^{ltr}, L_k^{lt}, L_k^{trb}, L_k^{trbl}, L_k^{blt} L_k^{rb}, L_k^{rbl}$, and L_k^{bl} can be determined.

3.2 The 1-D Leap-Frog

The *1-D Leap-Frog Algorithm* (see Noakes [1]) applies to a class of nonlinear problems and 2-D Leap-Frog is a 2-dimensional construction of the same sort. The 1-D Leap-Frog shows how these kinds of constructions are sometimes able to handle nonlinearities. It is enough to focus on the following special case.

Let $\mathbb{R}^2$ be equipped with some Riemannian metric and let $x_0, x_1 \in \mathbb{R}^2$. The length $\lambda(\gamma)$ of a C^1 curve $\gamma : [0,1] \to \mathbb{R}^2$ is defined with respect to the Riemannian metric. Suppose from now on that $\gamma(0) = x_0, \gamma(1) = x_1$ and that γ is parameterized proportionally to arc-length. The functional λ acts on the infinite dimensional space of all such curves γ, and critical points of λ are *geodesics*, namely they satisfy a particular form of the Euler-Lagrange equation in the calculus of variations (see Klingenberg [25, Chapter 5]).

In order to calculate *Riemannian distances* we usually need to find a geodesic joining x_0 and x_1, which is a *boundary value problem* for the geodesic equations. Such problems can be solved by *shooting methods* (see Keller [26]) which work well when a good initial guess is available, especially when points x_0 and x_1 are nearby. The 1-D Leap-Frog works with no need for a good initial guess Γ_1 of Γ_{crit} (though the better Γ_1 is the quicker convergence is achieved).

The algorithm follows the pattern:

- select an initial curve Γ_1 of Γ_{crit} joining x_0 and x_1 (see Figure 1 (a)). An estimate might be a straight line passing through both points. Divide Γ_1 into, say, four subpaths each joining the pairs of points: (x_0, x_1^1), (x_1^1, x_2^1), (x_2^1, x_3^1), and (x_3^1, x_1).

- using a shooting method (applied now to a more local problem) find the geodesic path between points x_0 and x_2^1. Select the middle point x_1^2 on this path and then by using shooting method find the geodesic path between

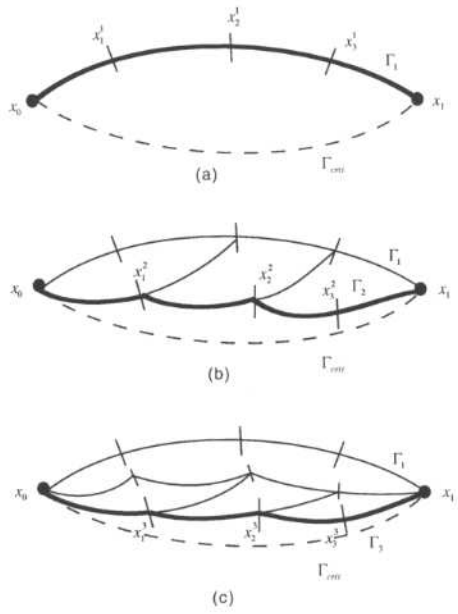


Figure 1: 1-D Leap-Frog Algorithm yielding approximations Γ_n to a critical path Γ_{crit} with: (a) an initial estimate Γ_1 (b) the second estimate Γ_2 (c) the third estimate Γ_3.

x_1^2 and x_3^1. Similarly, choose the middle point x_2^2 on that path and using shooting method find the geodesic path joining x_2^2 with x_1. Finally, localize and mark the middle point x_3^2 on that last subpath. The sum of subpaths joining pairs of points (x_0, x_1^2) (x_1^2, x_2^2), (x_2^2, x_3^2) and (x_3^2, x_1) constitutes now an updated estimate Γ_2 of Γ_{crit} (see Figure 1 (b)).

- continue the next iterations yielding Γ_n $(n \geq 2)$ as specified in the previous step (Γ_3 is shown in Figure 1 (c)).

It can be shown that, subject to certain conditions, the sequence of curves Γ_n converges to Γ_{crit}. The idea is to exploit the power of shooting methods in finding a local optimum, and then somehow merge together subotptimal solutions. This also forms the conceptual back-bone of the 2-D Leap-Frog. Further background information can found in Noakes [1, 27] and Kaya and Noakes [28].

The 1-D Leap-Frog:

1. solves a global problem,
2. is iterative,
3. converges regardless of the initial guess,
4. works within the space of feasibile objects,
5. has each step relatively small-scale, allowing for rapid computation,

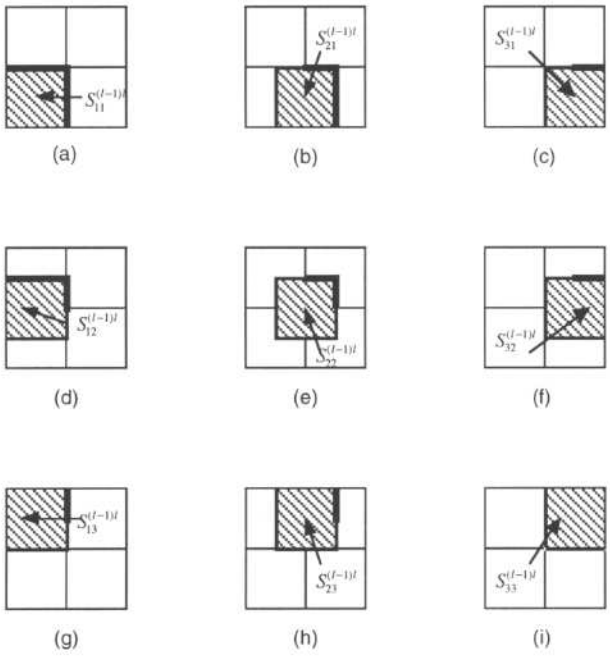


Figure 2: Covering an image Ω by the family $\mathcal{F}^{kl} = \{S_{ij}^{(l-1)l}\}_{1 \le i,j \le 3}$ of subsquares (here $k = l-1$). Each $S_{ij}^{(l-1)l}$ consists of $2^{2(l-1)}$ atomic squares.

6. is amenable to the parallelism, and

7. deals with nonlinearities.

These features are observable for 2-D Leap-Frog also, except that in the present paper the method is not required to deal with nonlinearities.

3.3 2-D Leap-Frog

To describe 2-D Leap-Frog, suppose given a nonintegrable vector field $\vec{v}$ defined over a rectangular grid in $\Omega = [0,1] \times [0,1]$ with 2^l sides. The optimization problem (7) is solvable in principle by the methods of section 2, but in practice this is unworkable by direct methods unless l is small. Certainly $l \ge 7$ makes diffficulties. So the following algorithm uses a sequence of optimizations where l is replaced by a smaller integer k.

Cover Ω by a familiy of overlapping subsquares $\mathcal{F}^{kl} = \{S_{ij}^{kl}\}_{1 \le i,j \le 2^{l-k+1}-1}$ each comprising 2^{2k} atomic subsquares $S_{i_q j_g}^{l}$. Each S_{ij}^{kl} is defined as follows $S_{ij}^{kl} = [(i-1)2^{k-l-1}, (i-1)2^{k-l-1} + 2^{k-l}] \times [(j-1)2^{k-l-1}, (j-1)2^{k-l-1} + 2^{k-l}]$, where $1 \le i,j \le 2^{l-k+1} - 1$. The case when $k = l-1$ (and $l \ge 2$ is arbitrary) is shown in Figure 2. Let $\hat{\vec{v}}^1$ be any integrable vector field. Many choices are possible. For example we might use the methods of Noakes, Kozera and Klette [2] or Frankot and Chellappa [3] or Horn [4] to approximate $\hat{\vec{v}}$. The easiest (and perhaps best)

choice is $\hat{\vec{v}}^1 = \vec{0}$. We present here the 2-D Leap-Frog version with half S_{ij}^{kl} overlaps. The size of the overlaps can vary (between 1 and $2^k - 1$ pixels) as well as the size of the snapshot S_{ij}^{kl}, where $1 \leq k \leq l-1$. Theorem 1 can be proved for these cases also, by the same argument. For $n = 1, 2, \ldots$ repeat the following sequence of steps until some halting condition is flagged (bounds for the number of iterations, estimated error, or deficiency angle as in section 5).

- start with the left bottom subsquare S_{11}^{kl} and apply a least-square optimization with respect to[2] $\vec{v}$, with fixed top-right boundary constraints inherited from the values computed in the $n-1$th iteration (for $k = l-1$ see Figure 2 (a)).

- pass now to the second subsquare of the first row S_{21}^{kl} and apply a least-square optimization with fixed left-top-right boundary constraints. The left and left-half top boundary conditions are inherited from computed values over S_{11}^{kl} obtained in the nth iteration. The right and right-half top boundary conditions are inherited from the computed values in obtained $n-1$th iteration (for $k = l-1$ see Figure 2 (b)).

- this process continues until the last subsquare $S_{(2^{l-k+1}-1)1}^{kl}$ in the first row. Over this subsquare apply a least-square optimization with left-top boundary constraints fixed. The left and left-half top boundary conditions are given by the previously computed values over $S_{(2^{l-k+1}-2)1}^{kl}$ in nth iteration. The right-half top boundary conditions are given by the values obtained in the $n-1$th iteration (for $k = l-1$ see Figure 2 (c)). Thus the first row of nth iteration is completed.

- next pass to the second row. Start with the S_{12}^{kl} subsquare and apply a least-square optimization with fixed top-right-bottom boundary constraints. The bottom and bottom-half right boundary constraints are inherited from nth iteration values computed over subsquares S_{11}^{kl}, S_{21}^{kl} and S_{31}^{kl}. The top and top-half right boundary conditions are inherited from values computed in the $n-1$th iteration (for $k = l-1$ see Figure 2 (d)).

- pass to the second subsquare S_{22}^{kl} over which we apply a least-square optimization with top-right-bottom-left boundary constraints. The left-half top, the left, the bottom and the bottom-half right boundary conditions are inherited from the values of the nth iteration computed over S_{12}^{kl}, S_{31}^{kl} and S_{41}^{kl}. The remaining boundary conditions are inherited from computed values obtained in the $n-1$th iteration (for $k = l-1$ see Figure 2 (e); note that in case when $k = l-1$ there is no dependence on S_{41}^{kl}-boundary).

- continue until the last subsquare $S_{(2^{l-k+1}-1)2}^{kl}$ in the second row is reached. Over this subsquare apply a least-square optimization with bottom-left-top

[2]More precisely, adjust the variables in the snapshot to minimize the distance to the given nonintegrable field $\vec{v}$. All optimizations in 2-D Leap-Frog are to be understood in this sense.

boundary constraints fixed. The left-half top, the left, and the bottom boundary conditions are inherited from the values computed in the nth iteration over $S^{kl}_{(2^{l-k+1}-2)2}$ and $S^{kl}_{(2^{l-k+1}-1)1}$. The right-half top boundary conditions are inherited from values computed in the $n-1$th iteration. Thus the second row of the nth iteration is completed (for $k = l-1$ see Figure 2 (f)).

- continue row by row (as specified in the previous step), until the last row is reached. Apply now a least-square optimization over $S^{kl}_{12^{l-k+1}-1}$ with fixed right and bottom boundary constraints. The bottom and the bottom-half right boundary conditions are inherited from values computed in the nth iteration over $S^{kl}_{12^{l-k+1}-2}$, $S^{kl}_{22^{l-k+1}-2}$, and $S^{kl}_{32^{l-k+1}-2}$. The top-half right boundary conditions are inherited from values computed in the $n-1$th iteration (for $k = l-1$ see Figure 2 (g)).

- pass to the second subsquare of the last row $S^{kl}_{22^{l-k+1}-1}$ over which we apply a least-square optimization with right-bottom-left boundary constraints fixed. The bottom-half right, the bottom and the left boundary conditions are inherited from values of the nth iteration computed over $S^{kl}_{12^{l-k+1}-1}$, $S^{kl}_{32^{l-k+1}-2}$ and $S^{kl}_{42^{l-k+1}-2}$. The top-half right boundary conditions are inherited from computed values obtained in the $n-1$th iteration (for $k = l-1$ see Figure 2 (h); note that when $k = l-1$ there is no dependence on $S^{kl}_{42^{l-k+1}-2}$-boundary).

- this process continues, up until the last subsquare $S^{kl}_{(2^{l-k+1}-1)(2^{l-k+1}-1)}$, in the last row is reached. Over this subsquare apply a least-square optimization with the bottom-left boundary constraints fixed and inherited from computed values over $S^{kl}_{(2^{l-k+1}-2)(2^{l-k+1}-1)}$ and $S^{kl}_{(2^{l-k+1}-1)(2^{l-k+1}-2)}$ in the nth iteration (for $k = l-1$ see Figure 2 (i)). This completes the nth iteration, and the resulting integrable vector field is labelled $\hat{\vec{v}}^{n+1}$.

Evidently, a generic snapshot optimization performed during the execution of 2-D Leap-Frog deals with the case when all boundary constraints are fixed, as it happens with S^{kl}_{22} for example.

4 Convergence of 2-D Leap-Frog

This section is devoted to the proof of linear convergence of the 2-D Leap-Frog, namely the following.

Theorem 1 *Let $\hat{\vec{v}}$ be the optimal integrable vector field for the problem (7) described in section 3. Then for some constant $C \in [0,1)$ and for each $n \in \mathbb{N}$*

$$\|\hat{\vec{v}}^{n+1} - \hat{\vec{v}}\| \leq C\|\hat{\vec{v}}^{n} - \hat{\vec{v}}\|. \tag{8}$$

Note that as usual formula (8) implies (linear) convergence of 2-D Leap-Frog to the optimum $\hat{\vec{v}}$ (which coincides with the orthogonal projection of $\vec{v}$ onto the subspace of integrable vector fields).

Preliminaries: To prove Theorem 1 (but not to implement 2-D Leap-Frog) it is necessary to change the way we look at the $\hat{\vec{v}}^n$. For purposes of proof it is rather unhelpful to represent these as vectors in $\mathbb{R}^{2^{l+1}(2^l+1)}$. It is better to view them as abstract vectors or *tableaux* as we shall call them. A *tableau* is an array of real numbers of the form

$$v_T = \left(\begin{array}{ccccccccccc}
 & x_{2^{2l}+1} & & x_{2^{2l}+2} & & \cdots & \cdots & & x_{2^l(2^l+1)} & & \\
y_{2^l} & \square & y_{2^{2l}} & \square & y_{3^{2l}} & & \cdots & y_{2^{2l}} & \square & y_{2^l(2^l+1)} & \\
 & x_{2^{2l-1}+1} & & x_{2^{2l-1}+2} & & \cdots & \cdots & & x_{2^{2l}} & & \\
\vdots & & \vdots & & \vdots & & & \vdots & & \vdots & \\
 & \vdots & & \vdots & & & & & \vdots & & \\
\vdots & & \vdots & & \vdots & & & \vdots & & \vdots & \\
 & x_{2^{2l}+1} & & x_{2^{2l}+2} & & \cdots & \cdots & & x_{3^{2l}} & & \\
y_2 & \square & y_{2^l+2} & \square & y_{2^{2l}+2} & & \cdots & y_{2^{2l-1}+2} & \square & y_{2^{2l}+2} & \\
 & x_{2^l+1} & & x_{2^l+2} & & \cdots & \cdots & & x_{2^{2l}} & & \\
y_1 & \square & y_{2^l+1} & \square & y_{2^{2l}+1} & & \cdots & y_{2^{2l-1}+1} & \square & y_{2^{2l}+1} & \\
 & x_1 & & x_2 & & \cdots & \cdots & & x_{2^l} & &
\end{array} \right). \tag{9}$$

Even without the $\square$ symbols, a tableau is not a matrix (neither rows nor columns have the same number of entries). A $\square$ symbol in a tableau is called a *nucleus* and its adjacent real entries are its *neighbours*. Given l, the set of all tableaux is denoted by V_T.

For $v_{1T}, v_{2T} \in V_T$ define $v_{1T} +_T v_{2T} \stackrel{\text{def}}{=} v_T$ by $x_s^{v_T} = x_s^{v_{1T}} + x_s^{v_{2T}}$ and $y_s^{v_T} = y_s^{v_{1T}} + y_s^{v_{2T}}$. For $v_T \in V_T$ and $\mu \in \mathbb{R}$, define $w_T \stackrel{\text{def}}{=} \mu \cdot_T v_T$ by $x_s^{w_T} = \mu x_s^{v_T}$ and $y_s^{w_T} = \mu y_s^{v_T}$. With respect to these operations of vector addition and scalar multiplication, $(V_T, +_T, \cdot_T)$ is a vector space, equipped with a natural isomorphism $\mathcal{I}$ to $\mathbb{R}^{2^{l+1}(2^l+1)}$. Define an inner product $< \cdot \mid \cdot >_T$ on V_T by multiplying corresponding real entries in two tableaux and then summing. Denote the associated norm by $\| \cdot \|_T$. Then $\mathcal{I}$ is an isometry to Euclidean space $\mathbb{R}^{2^{l+1}(2^l+1)}$.

A tableau (9) is said to be *integrable* when, for all nuclei, the sum of its four neighbours is 0. More specifically, for an integrable tableau,

$$x_{m2^l+n} + y_{n2^l+(m+1)} - x_{(m+1)2^l+n} - y_{(n-1)2^l+(m+1)} = 0.$$

The integrable tableaux comprise a vector subspace $V_T^{\oplus}$ of the vector space V_T of all tableaux. The following lemma holds in the context of 2-D Leap-Frog as described in section 3.

Lemma 2 *There is an affine transformation $A_T : V_T^{\oplus} \to V_T^{\oplus}$ such that*

$$\hat{\vec{v}}_T^{n+1} = A_T \hat{\vec{v}}_T^{n}, \tag{10}$$

where $\hat{\vec{v}}_T^{k} = \mathcal{I}(\hat{\vec{v}}^k)$ (for $k \in \mathbb{N}$) denotes the representation (via $\mathcal{I}$) of $\hat{\vec{v}}^k$ in the space V_T.

Proof: Note that, in case of a half snapshot overlap, one iteration of 2-D Leap-Frog consists of $(2^{l+1-k})^2$ snapshot optimizers. Formula (5) shows that each snapshot optimizer forms an affine mapping $V_T^{\oplus} \to V_T^{\oplus}$. Indeed, looking at formula (5) yields that for a snapshot on a given location the vector $\vec{v}$ is a projection (in fact orthogonal) of $\vec{v}_h$ (see section 3.1). The term $\vec{u}$ in (5) is linear on any vector representing the integrable vector field. So the right-hand side of (5) is an affine on integrable vector fields (depending on the position of the snapshot). Thus 2-D Leap-Frog is an affine mapping $A(\hat{\vec{v}}^n) = A^L(\hat{\vec{v}}^n) + \hat{\vec{w}}$ from the subspace of integrable vector fields to itself; A^L forms its linear component. Hence the transformation

$$A_T(\hat{\vec{v}}_T^n) = \mathcal{I}(A(\mathcal{I}^{-1}(\hat{\vec{v}}_T^n))) = A_T^L(\hat{\vec{v}}_T^n) + \hat{\vec{w}}_T, \tag{11}$$

where $A_T^L = \mathcal{I} \circ A^L \circ \mathcal{I}^{-1}$ and $\mathcal{I}(\hat{\vec{w}}) = \hat{\vec{w}}_T$, yields (10). □

Note that in Lemma 2 the mapping A, and thus A_T are the same over each iteration of 2-D Leap-Frog.

Let W_M be the vector space of real $(2^l + 1) \times (2^l + 1)$ matrices. Think of the entries $\{a_{ij}\}$ of such a matrix as values of the function u calculated at grid points labelled (i, j), where $1 \leq i, j \leq 2^l + 1$. Define a linear transformation $\Phi : W_M \to V_T$ by mapping from function values to the corresponding discretised vector field, namely $\Phi(X) = Y$, where for $n = 2^l + 1$ we have

$$X = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & \dots & a_{1(n-1)} & a_{1n} \\ a_{21} & a_{22} & a_{23} & a_{24} & \dots & a_{2(n-1)} & a_{2n} \\ a_{31} & a_{32} & a_{33} & a_{34} & \dots & a_{3(n-1)} & a_{3n} \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ a_{(n-1)1} & a_{(n-1)2} & a_{(n-1)3} & a_{(n-1)4} & \dots & a_{(n-1)(n-1)} & a_{(n-1)n} \\ a_{n1} & a_{n2} & a_{n3} & a_{n4} & \dots & a_{n(n-1)} & a_{nn} \end{pmatrix}$$

and

$$Y = \begin{pmatrix} & a_{12} - a_{11} & & \dots & a_{1n} - a_{1(n-1)} & \\ a_{11} - a_{21} & \square & a_{12} - a_{22} & \dots & \square & a_{1n} - a_{2n} \\ & a_{22} - a_{21} & & \dots & a_{2n} - a_{2(n-1)} & \\ \vdots & & \vdots & & & \vdots \\ & \vdots & & & \vdots & \\ \vdots & & \vdots & & & \vdots \\ & a_{(n-1)2} - a_{(n-1)1} & & \dots & a_{(n-1)n} - a_{(n-1)(n-1)} & \\ a_{(n-1)1} - a_{n1} & \square & a_{(n-1)2} - a_{n2} & \dots & \square & a_{(n-1)n} - a_{nn} \\ & a_{n2} - a_{n1} & & \dots & a_{nn} - a_{n(n-1)} & \end{pmatrix}.$$

Notice that Φ maps into the space of integrable tableaux, namely

$$\Phi(W_M) \subset V_T^{\oplus} \subset V_T.$$

It is easily verified that the kernel of Φ is spanned by the constant tableau whose real entries are everywhere 1. Consequently Φ has rank $(2^l + 1)^2 - 1 = 2^l(2^l + 2)$.

This is the dimension of $V_T^{\oplus}$. So the image under Φ of any $(2^l+1)^2-1$ linearly independent matrices W_M constitutes a basis of $V_T^{\oplus}$. In particular, define

$$\mathcal{B}_T^{\oplus} = \{h_{11}^{\oplus}, h_{12}^{\oplus}, \ldots, h_{1n}^{\oplus}, h_{21}^{\oplus}, h_{22}^{\oplus}, \ldots, h_{2n}^{\oplus}, \ldots, h_{n1}^{\oplus}, h_{n2}^{\oplus}, \ldots, h_{n(n-1)}^{\oplus}\},$$

where

$$h_{11}^{\oplus} = \begin{pmatrix} & 0 & & 0 & \ldots & 0 & \\ 0 & & 0 & & \ldots & & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & & 0 & & \ldots & & 0 \\ & 0 & & 0 & \ldots & 0 & \\ x & & 0 & & \ldots & & 0 \\ & x & & 0 & \ldots & 0 & \end{pmatrix}, \qquad h_{12}^{\oplus} = \begin{pmatrix} & 0 & & 0 & \ldots & 0 & \\ 0 & & 0 & & \ldots & & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & & 0 & & \ldots & & 0 \\ & 0 & & 0 & \ldots & 0 & \\ 0 & & x & & \ldots & & 0 \\ & y & & x & \ldots & 0 & \end{pmatrix}, \quad \ldots,$$

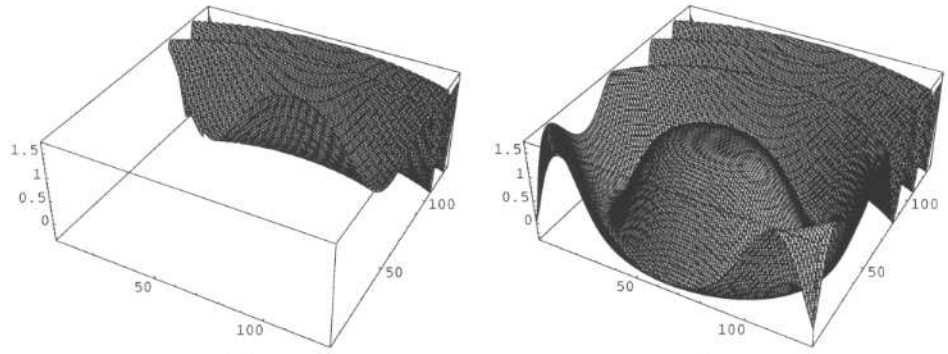
$$h_{1n}^{\oplus} = \begin{pmatrix} & 0 & & 0 & \ldots & 0 & \\ 0 & & 0 & & \ldots & & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & & 0 & & \ldots & & 0 \\ & 0 & & 0 & \ldots & 0 & \\ 0 & & 0 & & \ldots & & x \\ & 0 & & 0 & \ldots & y & \end{pmatrix}, \qquad h_{21}^{\oplus} = \begin{pmatrix} & 0 & & 0 & \ldots & 0 & \\ 0 & & 0 & & \ldots & & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ x & & 0 & & \ldots & & 0 \\ & x & & 0 & \ldots & 0 & \\ y & & 0 & & \ldots & & 0 \\ & 0 & & 0 & \ldots & 0 & \end{pmatrix},$$

$$h_{22}^{\oplus} = \begin{pmatrix} & 0 & & 0 & \ldots & 0 & \\ 0 & & 0 & & \ldots & & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & & x & & \ldots & & 0 \\ & y & & x & \ldots & 0 & \\ 0 & & y & & \ldots & & 0 \\ & 0 & & 0 & \ldots & 0 & \end{pmatrix}, \quad \ldots, \quad h_{2n}^{\oplus} = \begin{pmatrix} & 0 & & 0 & \ldots & 0 & \\ 0 & & 0 & & \ldots & & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & & 0 & & \ldots & & x \\ & 0 & & 0 & \ldots & y & \\ 0 & & 0 & & \ldots & & y \\ & 0 & & 0 & \ldots & 0 & \end{pmatrix},$$

and so on until

$$h_{n1}^{\oplus} = \begin{pmatrix} & x & & 0 & \ldots & 0 & \\ y & & 0 & & \ldots & & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \\ 0 & & 0 & & \ldots & & 0 \\ & 0 & & 0 & \ldots & 0 & \\ 0 & & 0 & & \ldots & & 0 \\ & 0 & & 0 & \ldots & 0 & \end{pmatrix}, \qquad h_{n2}^{\oplus} = \begin{pmatrix} & y & & x & \ldots & 0 & \\ 0 & & y & & \ldots & & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & & 0 & & \ldots & & 0 \\ & 0 & & 0 & \ldots & 0 & \\ 0 & & 0 & & \ldots & & 0 \\ & 0 & & 0 & \ldots & 0 & \end{pmatrix}, \quad \ldots,$$

$$h_{n(n-1)}^{\oplus} = \begin{pmatrix} & 0 & \ldots & y & & x & \\ 0 & & \ldots & & y & & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & & \ldots & & 0 & & 0 \\ & 0 & \ldots & 0 & & 0 & \\ 0 & & \ldots & & 0 & & 0 \\ & 0 & \ldots & 0 & & 0 & \end{pmatrix}.$$



(a) (b)

Figure 3: Graphs of the functions forming test surfaces for the shape reconstruction: (a) a quadratic polynomial $u_1(x,y) = x^2 + 3xy + 2y^2$ (b) a cosine wavy function $u_2(x,y) = \cos(20((x-0.5)^2 + 2(y-0.3)^2))$.

Then $\mathcal{B}_T^{\oplus}$ is a basis of the space $V_T^{\oplus}$ of integrable tableaux.

Note that the order of snapshots S_{ij}^{kl} taken in 2-D Leap-Frog corresponds geographically to the basis element of $\mathcal{B}_T^{\oplus}$ with the transposed indices *i.e.* to $h_{ji}^{\oplus}$.

Let $\vec{v}_T \in V_T$ be a nonintegrable tableau corresponding to the noisy data for problem (7), and define for $\hat{\vec{x}}_T \in V_T^{\oplus}$ the function $f : \mathbb{R}^s \to \mathbb{R}$ (here $s = 2^l(2^l+2)$), called *performance index* such that, for $n = 2^l + 1$, we have

$$f(x) = f(x_{11}, x_{12}, \ldots, x_{n(n-1)}) = \| \sum_{\substack{1 \le i,j \le n \\ (i,j) \ne (n,n)}} x_{ij} h_{ij}^{\oplus} - \vec{v}_T \|_T^2 = \bar{f}(\hat{\vec{x}}_T), \tag{12}$$

where $\| \cdot \|_T$ is computed from the inner product $< \cdot \mid \cdot >_T$ on V_T and $x_{ij} \in \mathbb{R}$. Then $x_o \in \mathbb{R}^{2^l(2^l+2)}$ is a (for $k \in \mathbb{N}$) denotes the representation (via $\mathcal{I}$) critical point of f (*i.e.* $\nabla f(x_o) = \vec{0}$) when, for $\hat{\vec{x}} = \mathcal{I}^{-1}(\hat{\vec{x}}_T)$ and $\vec{v} = \mathcal{I}^{-1}(\vec{v}_T)$

$$\bar{f}(\hat{\vec{x}}_T) = \|\hat{\vec{x}}_T - \vec{v}_T\| = \|\mathcal{I}^{-1}(\hat{\vec{x}}_T) - \mathcal{I}^{-1}(\vec{v}_T)\|^2 = \|\hat{\vec{x}} - \vec{v}\|^2 \tag{13}$$

attains its (global) minimum at $\hat{\vec{x}}_{To}$. Note that $\hat{\vec{x}}_{To}$ is an orthogonal projection $\hat{\vec{v}}_T$ of $\vec{v}_T$ on $V_T^{\oplus}$.

The importance of $\mathcal{B}_T^{\oplus}$ for 2-D Leap-Frog is due to the following lemma.

Lemma 3 *The vector $\hat{\vec{x}}_T \in V_T^{\oplus}$ is not optimal if and only if 2-D Leap-Frog improves the performance index* (12).

Proof. Each step of 2-D Leap-Frog improves (or at least does not decrease) the performance index f by adjusting some variables. To see it, note that once a given snapshot S_{ij}^{kl} is reached, 2-D Leap-Frog improves generically internal values of S_{ij}^{kl} while keeping the others fixed. In a rare situation if the boundary values coincide with those corresponding to $\hat{\vec{x}}_o$ (and thus $\hat{\vec{x}}_{To}$) no change is made. Hence (7) cannot be globally increased. This combined with (13) and (12) yields the

performance index f as a non-increasing function with respect to each snapshot otimization step. Evidently 2-D Leap-Frog cannot improve the global optimum $\hat{\vec{x}}_o$ at any stage by changing a few of those variables. So 2-D Leap-Frog fixes the optimum $\hat{\vec{x}}_o$. On the other hand, if $\hat{\vec{x}}$ (and thus $\hat{\vec{x}}_T$) is not already optimal, the gradient of f at the corresponding point in $v \in \mathbb{R}^{2^l(2^l+2)}$ is nonzero. Let (a,b) (here $1 \leq a,b \leq n$ and $(a,b) \neq (n,n)$, $n = 2^l + 1$) be the first indices when $(\partial f/\partial x_{ab})(v) \neq 0$ and when reached by step (b,a) in 2-D Leap-Frog, where the corresponding basis element $h^{\oplus}_{ba}$ is in the interior of the snapshot (recall here the geographical dependence between the order of 2-D Leap-Frog snapshots and the labelling of $h^{\oplus}_{ij}$). Since performance index f can be improved by changing only one variable v_{ab} the optimum when free to change any variable in the snapshot will have a smaller semi-local performance index. The improvement feeds through to the global tableau.□

We are ready now to complete the proof of Theorem 1.

In doing so note that combining Lemma 2 and (11) with $A_T(\hat{\vec{v}}_T) = \hat{\vec{v}}_T$ (see Lemma 3) we arrive at

$$\begin{aligned}\|\hat{\vec{x}}^{n+1} - \hat{\vec{v}}\| &= \|\hat{\vec{x}}^{n+1}_T - \hat{\vec{v}}_T\|_T = \|A_T(\hat{\vec{x}}^n_T) - \hat{\vec{v}}_T\|_T \\ &= \|A^L_T(\hat{\vec{x}}^n_T - \hat{\vec{v}}_T)\|_T \leq \|A^L_T\|\|\hat{\vec{x}}^n_T - \hat{\vec{v}}_T\|_T = C\|\hat{\vec{x}}^n - \hat{\vec{v}}\|,\end{aligned}$$

where $C = \|A^L_T\|$. It suffices to show that constant $C < 1$.

By Lemma 3 if $\hat{\vec{x}} \neq \hat{\vec{v}}$ then $\|A_T(\hat{\vec{x}}_T) - \vec{v}_T\|_T < \|\hat{\vec{x}}_T - \vec{v}_T\|_T$, and furthermore by Pythagoras' theorem

$$\|A_T(\hat{\vec{x}}_T) - \hat{\vec{v}}_T\|_T < \|\hat{\vec{x}}_T - \hat{\vec{v}}_T\|_T. \tag{14}$$

Let now $S_T(\hat{\vec{v}}_T, 1)$ be the unit sphere in the space $V^{\oplus}_T$ of integrable tableaux centered on the optimum $\hat{\vec{v}}_T$.

Defining $g : S_T(\hat{\vec{v}}_T, 1) \to \mathbb{R}$

$$g(\hat{\vec{x}}_{TS}) = \|A^L_T(\hat{\vec{x}}_{TS} - \hat{\vec{v}}_T)\|_T$$

and taking into account (14) we get for $\hat{\vec{x}}_{TS} \in S_T(\hat{\vec{v}}_T, 1)$

$$g(\hat{\vec{x}}_{TS}) = \|A_T(\hat{\vec{x}}_{TS}) - \hat{\vec{v}}_T\|_T < \|\hat{\vec{x}}_{TS} - \hat{\vec{v}}_T\|_T = 1. \tag{15}$$

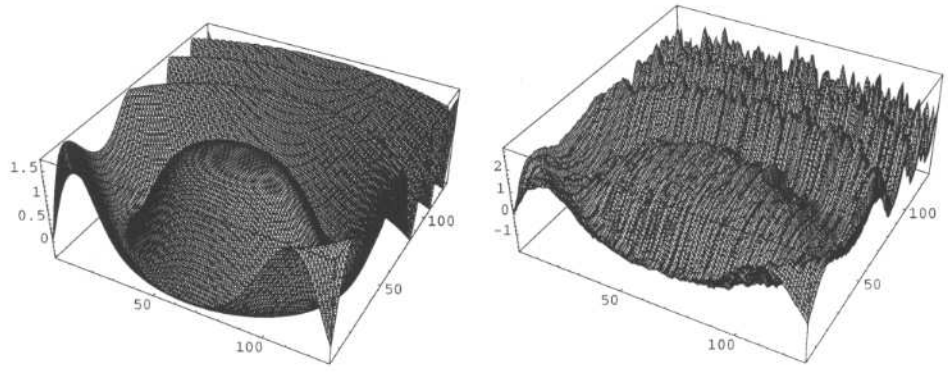
Note that g is defined on a compact set $S_T(\hat{\vec{v}}_T, 1)$ and thus attains its supremum for some $\hat{\vec{x}}^*{}_{TS} \in S_T(\hat{\vec{v}}_T, 1)$ which coupled with (15) yields

$$\sigma = sup\{g(\hat{\vec{x}}_{TS}) : \hat{\vec{x}}_{TS} \in S_T(\hat{\vec{v}}_T, 1)\} = g(\hat{\vec{x}}^*{}_{TS}) < 1.$$

The latter combined with the fact that $\sigma = \|A^L_T\|$ guarantees that $C < 1$. □

5 Experimentation and Conclusions

We now first briefly outline the performance of 2-D Leap-Frog. The tests were run in Mathematica on a PC 233MHz Pentium 2 with 64Mb RAM. We took



(a) (b)

Figure 4: The integrated surfaces (x, y, u_i^c) (for $i = 1, 2$) from the perturbed vector fields $\vec{v}_i$: (a) for the quadratic polynomial (b) for the cosine wavy function.

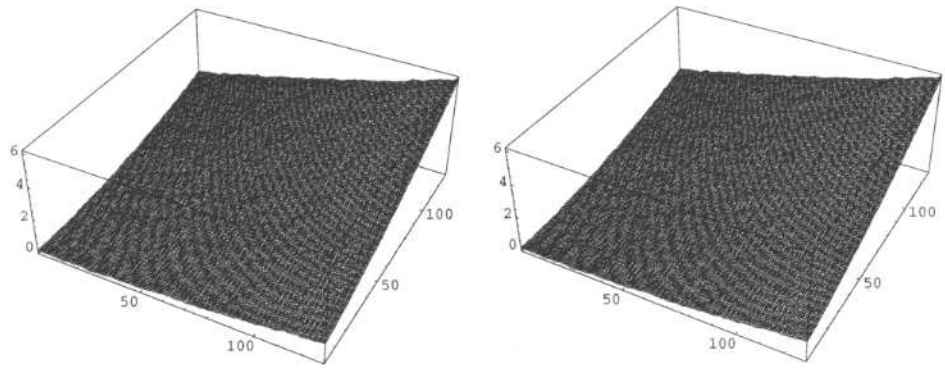
$k = 4$ and $l = 7$, *i.e.* each subsquare S_{ij}^{kl} consists of 16×16 pixel resolution and an image Ω consists of 128×128 pixels.

In generating $\vec{v}$ synthetically Gaussian noise was added, with mean zero and standard deviation 0.04, to the integrable gradient vector field $\vec{v}_{grad} = (u_x, u_y)$ obtained from the function $u_i : [0,1]\times[0,1] \to \mathbb{R}$ (for $i = 1, 2$) defined as $u_1(x, y) = x^2 + 3xy + 2y^2$ and $u_2(x, y) = \cos(20((x-0.5)^2 + 2(y-0.3)^2))$. The corresponding graphs of quadratic polynomial u_1 and cosine wavy function u_2 are illustrated in Figure 3. The standard trapezoidal integration scheme applied to (3), follows first the horizontal and then vertical path γ_{hv} joining $(x_0, y_0) = (0, 0)$ with an arbitrary image point $(x, y) \in \Omega$. Upon integration of the contaminated vector fields $\vec{v}_i$ (for $i = 1, 2$) the corresponding reconstructed surfaces $(x, y, u_1^c(x, y))$ and $(x, y, u_2^c(x, y))$ are plotted in Figure 4.

The 2-D Leap-Frog has been subsequently applied with an *a priori* iteration bound set to $n_0 = 21$. Lawn-Mowing was applied in the first iteration, *i.e.* $\hat{\vec{v}}^1 = \hat{\vec{v}}_{LM}$, followed by $n_0 - 1 = 20$ pure 2-D Leap-Frog iterations. For evaluation we introduce now a few auxialiary notations. Given the estimate $\hat{\vec{v}}^{n_0}$ of $\vec{v}$ obtained after 21 iterations define now $e = \|\vec{v} - \vec{v}_{grad}\|_2^2$, $c = \|\vec{v} - \hat{\vec{v}}^{n_0}\|_2^2$, $d = \|\vec{v}_{grad} - \hat{\vec{v}}^{n_0}\|_2^2$. Pythagoras' theorem applied to the triangle $\Delta(\vec{v}, \hat{\vec{v}}^{n_0}, \vec{v}_{grad})$ yields the cosine of the angle between sides $\overline{\vec{v}\hat{\vec{v}}^{n_0}}$ and $\overline{\vec{v}_{grad}\hat{\vec{v}}^{n_0}}$ expressed as:

$$\cos(\alpha) = \frac{d + c - e}{2\sqrt{dc}}.$$

Clearly, convergence of $\hat{\vec{v}}^n$ to $\hat{\vec{v}}$ translates into $\lim_{n\to\infty} \cos(\alpha_n) = 0$. Denoting $\beta_n = \frac{\pi}{2} - \arccos(\alpha_n)$ as the *angle deficiency*, expressed in radians, upon completion of nth iteration of the 2-D Leap-Frog (β_n should be close to zero) the following experimental results (each five-multiple iteration onwards) have been obtained:



(a) (b)

Figure 5: The integrated surfaces $(x, y, u_i^{n_0})$ (for $i = 1, 2$) by using 2-D Leap-Frog with $k = 4$ and $l = 7$: (a) for the quadratic polynomial (b) for the cosine wavy function.

Iteration No.	$n = 1$ Lawn-M.	$n = 2$	$n = 6$	$n = 11$	$n = 16$	$n_0 = 21$
β_n^1 for u_1	0.04256	0.00298	0.00050	0.00028	0.00018	0.00013
β_n^2 for u_2	0.03786	0.00286	0.00081	0.00033	0.00018	0.00011

The reconstructed surfaces $(x, y, u_1^{n_0}(x, y))$ and $(x, y, u_2^{n_0}(x, y))$ are illustrated in Figure 5. The corresponding errors $u_i^c - u_i^{n_0}$ (for $i = 1, 2$) are plotted in Figure 6 (a) and (b), accordingly.

We close the paper with some observations.

- the 2-D Leap-Frog substantially improves the Lawn-Mowing and visibly converges to the global optimum. The above tests confirm that the angle deficiency improvement ratio β_0/β_{n_0} is formidable. For example, for u_1 and u_2, upon n_0 iterations, it is 327 and 344, respectively. Note also that the first pure 2-D Leap-Frog (the second iteration) as opposed to the initial Lawn-Mowing iteration offers a substantial improvement β_0/β_1 which in case of u_1 is 14 and for u_2 is 13.

- note also that the angle deficiency β, upon applying Lawn-Mowing (used as the first iteration of the 2-D Leap-Frog) is close to zero. Thus the suboptimal solution provided by the Lawn-Mowing might be sufficient should the need for finding the global optimum not be a driving criterion. The latter is important, as n_0 iterations of the 2-D Leap-Frog take approximately (due to different boundary constraints) $n_0(2^{l+1-k} - 1)^2/2^{l-k} \approx 4n_0$ times longer than Lawn-Mowing. Note that in the case of family $\mathcal{F}^{kl}$ defined in section 3, Lawn Mowing optimizes over a subfamily of 4 nonoverlapping subsquares. Though computational time is an issue, 2-D Leap-Frog is a useful algorithm convergent to the global optimum.

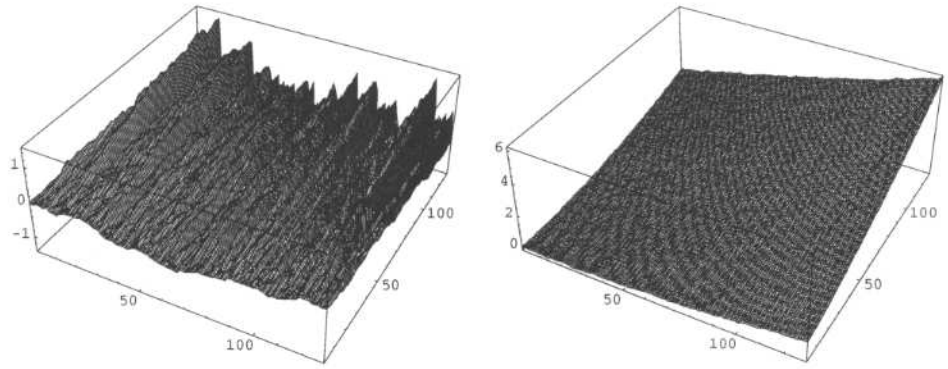
- splitting a global minimization problem into a collection of overlapping local minimization problems, in case of the 2-D and 1-D Leap-Frog Algorithm, is driven by essentially different reasons. Namely, in the latter case, shooting methods perform satisfactory, only if a local approach is used. Contrary to this, the least-square optimization method applied locally or globally solves perfectly (within the theoretical framework) integrability problem (7). However, once the issue of computational complexity is raised the advantage of resorting to the local approach (considered over each S_{ij}^{kl} with $k < l$) becomes more apparent. Indeed, the analysis contained in subsection 3.1 reveals, that for the pixel resolution $2^8 \times 2^8 = 256 \times 256$ a corresponding inhomogeneous system $L_8^h(\vec{x}) = \vec{0}$ with $k = l = 8$, comprises $2^{16} = 65,536$ equations in $2^9(2^8 + 1) = 131,584$ unknowns. The latter defines the matrix of 8,623,620,000 coefficients which for 8 byte precision numbers, requires about 70 GByte RAM. So direct global optimization constitutes an infeasible task. There exist many methods exploiting sparseness of the matrix LL^T and/or external memory but our method is essentially geometrical and well adapted to the nature of global and local integrability problem encapsulated in formula (7).

- the satisfaction of the discrete integrability condition (6) assumes the linear interpolation of a given function u, consistent with the trapezoidal method used later for refined vector field integration. Both illustrations in Figure 6 indicate the direction of the error propagation consistent with the horizontal-vertical path integration scheme.

- angle deficiency is used here merely as an algorithm evaluation indicator not as an *a posteriori* halting condition. The latter can also be used in setting the implicit upper bound for the number of iterations to halt once $\beta_n < \varepsilon$ (where ε represents the requested angle deficiency accuracy).

- any algorithm based on least-square optimization (including Lawn-Mowing and the 2-D Leap-Frog) cannot remove the entire noise from $\hat{\vec{v}}$ and thus is unable to retrieve the genuine $\vec{v}_{grad}$. The omission of the truncation error (treated as negligible) is only a minor contributor to the mentioned above property. Indeed, as Gaussian noise is added in all directions, any perturbation within the affine subspace A_L (comprising integrable vector fields, modulo truncation error) remains undetectable since then $\vec{v} = \hat{\vec{v}}$. The only detectable and removable noise component belongs to the orthogonal complement $A_L^{\perp}$. In the case of homogeneous system, assuming that $rank(L_k^h) = m$, the ratio of $(dim(ker(L_k^h)))/(2^{k+1}(2^k + 1))) = (2^k(2^k + 2))/(2^{k+1}(2^k + 1)) \approx 0.5$ indicates the proportion of the unremovable noise from the total noise incorporated in perturbed vector field $\vec{v}$. Similar ratios can be calculated for subsquares S_t with various boundary constraints discussed in subsection 3.1.

- like the nonlinear 1-D Leap-Frog, 2-D Leap-Frog is geometrical, and has



(a) (b)

Figure 6: The errors between u_i^c and $u_i^{n_0}$ (where $i = 1, 2$): (a) for the quadratic polynomial (b) for the cosine wavy function.

many other features in common (see 3.2). Inspection shows that the order in which snapshots are taken can be varied. Also there are opportunities for parallel processing.

- 2-D Leap-Frog is also related to classical algebraic methods in numerical analysis. In the special case $k = 1$ 2-D Leap-Frog is Gauss-Seidel for a system with respect to our canonical basis. Alternatively Gauss-Seidel could be applied directly to the larger system (5) with respect to the standard Euclidean basis. For $k > 1$ is multiplicative Schwarz (see Benzi, Nabben, Szyld [23]). See Hackbush [29] for other methods for sparse linear systems.

- in the geometrical proof of Theorem 1 single coefficient variation x_{ij} associated with $h_{ij}^{\oplus}$ changes only 4 points (or 3 or 2 if S_{ji}^{kl} coincides with the boundary snapshot) positioned inside a given snapshot. This is the reason for the use of our canonical basis $\mathcal{B}_T^{\oplus}$. Examining this argument, we see that the size of the half snapshot overlap can be varied between 1 and $2^k - 1$.

Acknowledgement

The authors thank Professor Maksymilian Dryja (Intitute of Applied Mathematics, Warsaw University) for helpful comments on aspects of this paper and useful references. We also thank Professor Gerald Sommer for useful conversations and references, and for arranging support by the Institute of Computer Science and Applied Mathematics at Christian-Albrecht University in Kiel, where a significant amount of work of this paper was carried out. Finally, the authors are grateful for the support from The Australian Research Council and The Alexander von Humboldt Foundation.

References

[1] L. Noakes, "A global algorithm for geodesics," *Journal of Mathematical Australian Society* **64**(Series A), pp. 37–50, 1999.

[2] L. Noakes, R. Kozera, and R. Klette, "The Lawn-Mowing Algorithm for noisy gradient vector fields," in SPIE'99 [30], pp. 305–316.

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

[4] B. K. P. Horn, "Height and gradient from shading," *International Journal of Computer Vision* **5**(1), pp. 37–75, 1990.

[5] L. Noakes and R. Kozera, "A 2-D Leap-Frog Algorithm for optimal surface reconstruction," in SPIE'99 [30], pp. 317–328.

[6] B. K. P. Horn, *Robot Vision*, McGraw-Hill, New York, Cambridge, MA, 1986.

[7] B. K. P. Horn and M. J. Brooks, eds., *Shape from Shading*, MIT Press, Cambridge, MA, 1989.

[8] M. J. Brooks and W. Chojnacki, "Direct computation of shape from shading," in *Proceedings of the Twelfth International Conference on Pattern Recognition*, pp. 114–119, IEEE Computer Society Press, Los Alamitos, CA, (Jerusalem, Israel, October 9–13), 1994.

[9] M. J. Brooks, W. Chojnacki, and R. Kozera, "Circularly symmetric eikonal equations and non-uniqueness in computer vision," *Journal of Mathematical Analysis and Applications* **165**(1), pp. 192–215, 1992.

[10] M. J. Brooks, W. Chojnacki, and R. Kozera, "Impossible and ambiguous shading patterns," *International Journal of Computer Vision* **7**(2), pp. 119–126, 1992.

[11] M. J. Brooks, W. Chojnacki, and R. Kozera, "Shape without shading," *Quarterly of Applied Mathematics* **50**(1), pp. 27–38, 1992.

[12] P. Deift and J. Sylvester, "Some remarks on the shape-from-shading problem in computer vision," *Journal of Mathematical Analysis and Applications* **84**(1), pp. 235–248, 1981.

[13] P. Dupuis and J. Oliensis, "An optimal control formulation and related numerical methods for a problem in shape reconstruction," *Annals of Applied Probability* **4**(2), pp. 278–346, 1994.

[14] R. Kimmel and A. Bruckstein, "Global shape from shading," *Computer Vision and Image Understanding* **62**(3), pp. 360–369, 1995.

[15] R. Klette, K. R. Schlüns, and A. Koschan, *Computer Vision - Three Dimensional Data from Images*, Springer, Singapore, 1998.

[16] R. Kozera, "Existence and uniqueness in photometric stereo," *Applied Mathematics and Computation* **44**(1), pp. 1–104, 1991.

[17] R. Kozera, "On shape recovery from two shading patterns," *International Journal of Pattern Recognition and Artificial Intelligence* **6**(4), pp. 673–698, 1992.

[18] R. Kozera, "Uniqueness in shape from shading revisited," *International Journal of Mathematical Imaging and Vision* **7**, pp. 123–138, 1997.

[19] R. Kozera, "A note on complete integrals and uniqueness in shape from shading," *Applied Mathematics and Computation* **73**(1), pp. 1–37, 1995.

[20] J. Oliensis, "Uniqueness in shape from shading," *International Journal of Computer Vision* **6**(2), pp. 75–104, 1991.

[21] R. Onn and A. Bruckstein, "Integrability disambiguates surface recovery in two-image photometric stereo," *International Journal of Computer Vision* **5**(1), pp. 105–113, 1990.

[22] E. Rouy and A. Tourin, "A viscosity solutions approach to shape-from-shading," *SIAM Journal of Mathematical Analysis* **29**, pp. 867–884, 1992.

[23] R. Benzi, M. Nabben and D. Szyld, "Algebraic theory of multiplicative Schwarz methods," Tech. Rep. 00210, Department of Mathematics, Temple University, Pliladelphia, USA, http : //www.math.temple.edu/ ~szyld, 2000.

[24] S. Zubrzycki, *Lectures in Probability Theory and Mathematical Statistics*, American Elsevier Publishing Company Inc., New York, 1972.

[25] W. Klingenberg, *A Course in Differential Geometry*, Springer-Verlag, New York, 1978.

[26] H. B. Keller, *Numerical Methods for Two-Point Boundary-Value Problems*, Blaisdell, 1968.

[27] L. Noakes, "Accelerations of Riemannian quadratics," *Proceedings of American Mathematical Society* **127**, pp. 1827–1836, 1999.

[28] C. Y. Kaya and L. Noakes, "A Leap-Frog Algorithm and optimal control: theoretical aspects," in *Proceedings of International Conference on Optimization Technics and Applications*, pp. 843–850, Curtin University of Technology, (Perth, Australia, July 1–3), 1998.

[29] W. Hackbush, *Iterative Solution of Large Sparse Systems of Equations*, Springer-Verlag, New York-Berlin-Heidelberg, 1994.

[30] *Proceedings of the 44th Annual Meeting of the Optical Engineering, stream Vision Geometry*, (Denver, Colorado, USA, July 19–20), IEEE Computer Society Press, 1999.