[15] Comparison of the Efficiency of Deterministic and Stochastic Algorithms for Visual Reconstruction

Andrew Blake

Department of Computer Science
University of Edinburgh, Edinburgh EH9 3JZ, Scotland, UK

Abstract

Piecewise continuous reconstruction of real-valued data can be formulated in terms of non-convex optimisation problems. Both stochastic and deterministic algorithms have been devised to solve them. The simplest such reconstruction process is the "weak string". Exact solutions can be obtained for it, and are used to determine the success or failure of the algorithms under precisely controlled conditions. It is concluded that the deterministic algorithm (Graduated Non-Convexity) outstrips stochastic (Simulated Annealing) algorithms both in computational efficiency and in problem-solving power. Piecewise continuous reconstruction of real-valued data can be formulated in terms of non-convex optimisation problems. Both stochastic and deterministic algorithms have been devised to solve them. The simplest such reconstruction process is the "weak string". Exact solutions can be obtained for it, and are used to determine the success or failure of the algorithms under precisely controlled conditions. It is concluded that the deterministic algorithm (Graduated Non-Convexity) outstrips stochastic (Simulated Annealing) algorithms both in computational efficiency and in problem-solving power.

1 Introduction

Visual Reconstruction is the reduction of noisy visual data to stable descriptions. An early stage in this process involves approximating data by continuous or piecewise continuous functions. In particular this paper is concerned with optimisation formulations for such tasks. Work in this area has included analysis of shading [29, 15, 13, 4], stereo [11, 27] and optic flow [14, 24]. More recently such methods have been extended to deal with discontinuities [3, 10, 22, 5, 6, 28, 23, 2, 8, 9, 18].

Both deterministic (relaxation) algorithms and stochastic ones (simulated annealing) have been used for visual reconstruction with discontinuities. Intuitively it might seem that stochastic algorithms, using random perturbations, should be less efficient than deterministic ones. We will show in carefully controlled comparisons that this is indeed the case.

The problem chosen for study is the "weak string" which is a 1D reconstruction process susceptible both to deterministic and stochastic algorithms. It is a means of approximating a noisy function $d(x)$ by a piecewise continuous function $u(x)$. It admits of an exact solution - an important property for benchmarking purposes. Note that $u(x)$ is real-valued, not restricted to a few levels or colours. This is an important point, as in many visual applications real-valued quantities are involved and must be estimated by a reconstruction process. Moreover the deterministic algorithm to be tested (GNC [3, 8]) does not lend itself to discrete valued problems.

Evaluation of Simulated Annealing in one particular problem [12] showed that it succeeds only if the characteristic "scale" or "smoothing parameter" of reconstruction is not too great. Another study [16] shows that the deterministic GNC algorithm requires about the same computational effort to solve the real-valued "weak membrane" problem as does Simulated Annealing to perform a similar but boolean-valued reconstruction. However the state-space for a real-valued problem is so much larger (i.e. uncountable) that it is reasonable to expect that more computational effort might be required than in the boolean case. This also accords with experience of deterministic algorithms [27, 8] in which increasing precision of reconstruction results in increased computational load. Those studies also reveal other important factors in the computational load. Computation time is strongly dependent both on scale of reconstruction and on the noise content of the data. Both factors will be examined in this paper.

Benchmarks used in the paper are for 1D reconstruction, using the weak string. Although results are for 1D data, there is some justification for the conjecture that they will apply to 2D - for example to the weak membrane whose computational structure is a direct 2D analogue of the weak string. Of course there exist phenomena in certain interaction models (e.g. phase transitions in Ising models) that occur only in 2D, not in 1D. However, relevant properties for piecewise continuous reconstruction (scale and sensitivity properties, resistance to noise, "gradient

*Current address: Dept. Engineering Science, Parks Rd., Oxford.

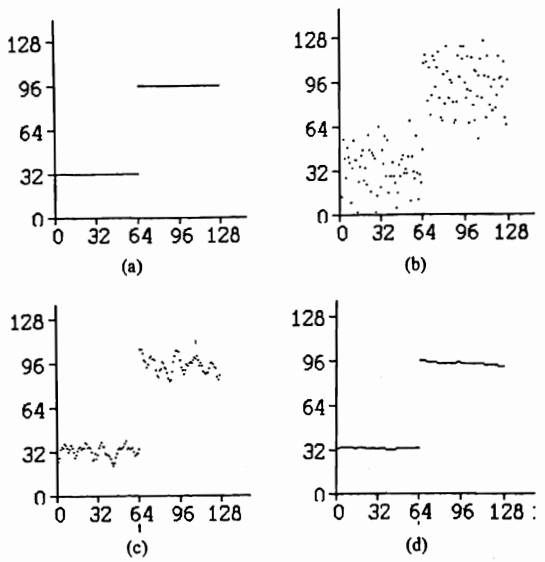


Figure 1: An isolated antisymmetric step (a) in random, uncorrelated gaussian noise of standard deviation 16 units (b) reconstructed by a weak string at small scale (c) and large scale (d).

limit") are common both to 1D and to 2D [8]. Moreover, performance of classical optimisation (i.e. with fixed rather than variable discontinuities) is known to be qualitatively similar in 1D and in 2D. And in the non-classical case, convergence of the GNC algorithm has been observed to be qualitatively similar both in 1D and in 2D.

The organisation of the paper is as follows. Section 2. defines the weak string problem and the algorithms to solve it, both deterministic and stochastic. Section 3. describes a new, exact dynamic programming solution for the weak string, to be used as an "assay" for benchmarks. Section 4. sets out results, using the benchmarks, of measurements of computational effort for deterministic and stochastic algorithms.

2 The weak string: problem and algorithms

In this section the weak string reconstruction problem is briefly described; a more detailed description is given in [8]. It is also more or less the simplest form of reconstruction for 1D signals that is capable of detecting and localising discontinuities. It is sufficiently simple that exact solutions can be computed (see next section) but sufficiently complex to be genuinely representative of a family of 1D and 2D reconstruction problems. An example of reconstruction by the weak string is given in figure 1.

2.1 The weak string

As with continuous reconstruction [11, 27] the weak string is defined in terms of functions and functionals. Given data $d(x)$ a reconstruction $u(x)$ is obtained by minimising an energy $E(u,d)$. This can be converted by means of "finite elements" [26, 30] to a discrete problem. A reconstruction

$$\mathbf{u} = \{u_i \; i = 1,..,N\}, \quad \mathbf{l} = \{l_i \; i = 1,..,N-1\}$$

is obtained from data $\mathbf{d} = \{d_i, \; i = 1,..,N\}$ by minimising the energy E:

$$\min_{u_i,l_i} E, \text{ where } E = D + S + P \tag{1}$$

and

$$\begin{aligned} D &= \sum_1^N (u_i - d_i)^2, \\ S &= \lambda^2 \sum_1^{N-1} (u_i - u_{i+1})^2 (1 - l_i), \\ P &= \alpha \sum_1^{N-1} l_i. \end{aligned} \tag{2}$$

The constant λ, the elasticity of the string, controls the scale of reconstruction. Constant α is a penalty levied for the inclusion of a breakpoint (discontinuity) and controls resistance to noise. Sensitivity is determined by the ratio $\sqrt{\alpha/\lambda}$. As it stands, the optimisation problem is "mixed" involving both boolean (l_i) and real (u_i) variables. It has been shown [8] that the mixed optimisation can be reduced to the following problem involving only real variables:

$$\min_{u_i} F \tag{3}$$

where

$$F = D + \sum_1^{N-1} g(u_i - u_{i+1}) \tag{4}$$

and

$$g(t) = \begin{cases} \lambda^2 t^2 & \text{if } |t| < \sqrt{\alpha}/\lambda \\ \alpha & \text{otherwise.} \end{cases} \tag{5}$$

Optimal values of l_i can be obtained explicitly from the optimal u_i as follows:

$$l_i = \begin{cases} 0 & \text{if } |u_i - u_{i+1}| < \sqrt{\alpha}/\lambda \\ 1 & \text{otherwise.} \end{cases} \tag{6}$$

The system can be understood by a mechanical analogy, in terms of coupled springs as in figure 2. It can also be understood in probabilistic terms as a "Markov Random Field" (MRF) whose prior probability density for a given state is simply $\exp(-(S+P)/T_0)$, where T_0 is a constant. A sample from the MRF is observed with additive gaussian noise whose probability density is $\exp(-D/T_0)$. The joint posterior probability of a particular set $\mathbf{d}$ of observed data is therefore proportional to

$$\exp(-(S+P)/T_0)\exp(-D/T_0) = \exp(-E/T_0). \tag{7}$$

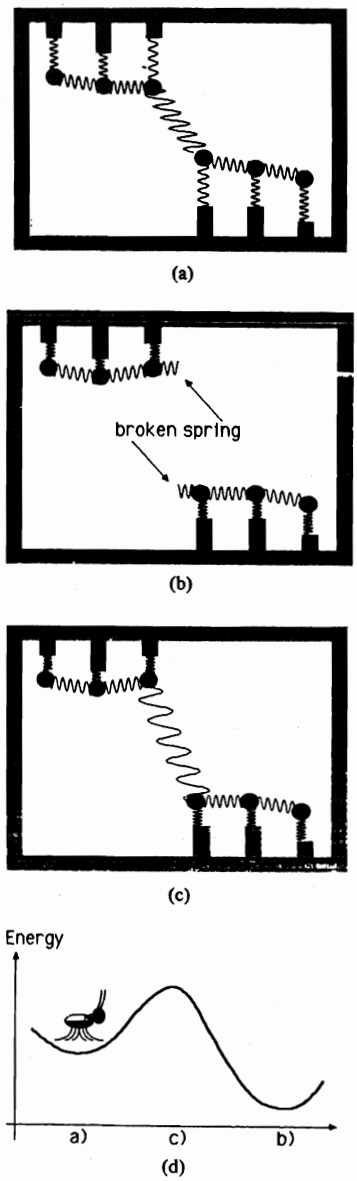


Figure 2: The weak string is like a system of conventional vertical springs with "breakable" lateral springs as shown. The states (a) and (b) are both stable, but the intermediate state (c) has higher energy than either (a) or (b). Suppose the lowest state is (b). A myopic fly with vertigo, crawling along the energy transition diagram (d), thinks state (a) is best. He has no way of seeing that, over the hump, he could get to a lower state (b). This is the "non-convexity" problem.

Minimising the energy E is therefore equivalent to maximising this posterior probability.

The energy $E(\mathbf{u}, \mathbf{l})$ is convex with respect to variables u_i, for fixed l_i, so that if line-variables l_i are fixed, classical optimisation procedures can be used to determine optimal u_i as in [11, 27]. But it is non-convex with respect to the variables l_i, so that when line-variables are treated as alterable classical optimisation is no longer adequate. (This is because non-convex functions may have many local minima; classical optimisation may lead to any one of them, which will not necessarily be a global minimimum.) Similarly, in the alternative form of the problem, $F(\mathbf{u})$ is non-convex with respect to the u_i - classical optimisation is no use there either. The purpose of this paper is to quantify the performance of algorithms which *are* capable of minimising some non-convex energies, including various stochastic algorithms and the deterministic "GNC" algorithm.

2.2 Stochastic optimisation

Stochastic algorithms for optimising non-convex energies have been described by various authors [25, 17, 10]. They use simulated annealing techniques in which the magnitude of successively applied random disturbances is controlled by a temperature parameter. Temperature (T) is lowered gradually according to a fixed "schedule". At high temperatures the system is able to jump out of local minima in the energy function and, as it cools, should settle into the system's ground state - its global energy minimum. The ground state can be achieved, in theory, [10] if a schedule is followed in which $T \propto 1/\log(n)$ (at the nth iteration or time-step of the system). Such a schedule takes an entirely unrealistic length of time. In practice a truncated logarithmic schedule is usually used but of course it can no longer be guaranteed that the ground state will be reached.

In this paper three variants of the simulated annealing algorithm for mixed variable problems are considered.

- The "Heatbath" (as described by Geman and Geman [10]) in which the thermal system is maintained in equilibrium as temperature decreases.
- What we will call the "Metropolis-Heatbath" algorithm in which the u_i are updated in the same way as in the Heatbath but the l_i are modified according to the Metropolis procedure [21].
- "Mixed annealing" [20] in which the u_i are updated according to a deterministic formula, but the l_i follow the Metropolis rule.

The first and last of these are included because they have been studied by other authors, and the Metropolis-Heatbath algorithm is interesting because it turns out to

be the most efficient of the three, at least for the weak string problem. The three algorithms will now be described briefly.

Heatbath

Each iteration consists of N visits made to randomly picked sites i, to update u_i and then l_i. Successive new values of u_i, l_i are generated by a "Gibbs Sampler" [10] - the values are chosen randomly from their conditional distibutions. Updating l_i is done by setting $l_i = l$ where l is picked randomly from the distribution

$$P_{l_i}(l) = P(l_i = l | u_j,\ j = 1,..,N;\ l_j,\ j = 1,..,N-1,\ j \neq i),$$

for $l \in \{0, 1\}$. For the weak string, this distribution is easily shown from (1), (2) and (7) to be

$$P_{l_i}(l) \propto \exp\left(-\frac{\alpha l + (1-l)(u_i - u_{i+1})^2\lambda^2}{T}\right). \quad (8)$$

Similarly u_i is updated to a value u chosen randomly from the distribution

$$P_{u_i}(u) = P(u_i = u | u_j,\ j = 1,..,N,\ j \neq i;\ l_j,\ j = 1,..,N-1).$$

For the weak string this is

$$P_{u_i}(u) \propto \exp\left(-\frac{(u-\mu_i)^2}{\sigma_i^2 T}\right) \quad (9)$$

where

$$\mu_i = \sigma_i^2\left(d_i + \lambda^2((1-l_{i-1})u_{i-1} + (1-l_i)u_{i+1})\right) \quad (10)$$

and

$$\sigma_i^2 = \left((2 - l_{i-1} - l_i)\lambda^2 + 1\right)^{-1} \quad (11)$$

These formulae are of course modified for sites near the ends $i = 1, N$. Temperature T is lowered according to a truncated logarithmic schedule

$$T = T_0\frac{\log(2)}{\log(2+n)}, \quad n \geq 0. \quad (12)$$

Metropolis-Heatbath

This algorithm works as the Heatbath except that line-variables l_i are updated according to the Metropolis procedure as follows. First calculate the energy change

$$\Delta E = E(u_1,..,u_N;\ l_1,..,l_{i-1}, 1-l_i, l_{i+1},..,l_{N-1}) - E(u_1,..,u_N;\ l_1,..,l_{i-1}, l_i, l_{i+1},..,l_{N-1})$$

which, for the weak string,

$$= (\alpha - (u_i - u_{i+1})^2\lambda^2)(1 - 2l_i). \quad (13)$$

Then do the following:

if	$\Delta E < 0$	**then**	$l_i \rightarrow 1 - l_i$
but if	$\Delta E \geq 0$	**then**	$l_i \rightarrow 1 - l_i$ with probability $\exp(-\Delta E/T)$.

That is, if energy would be decreased the change is invariably accepted. Otherwise the decision whether to make the change is made according to the toss of a (biassed) coin.

Mixed annealing

Line variables l_i are updated as in the Metropolis-Heatbath, but the u_i are updated deterministically as follows:

$$u_i \rightarrow (1-w)u_i + w\mu_i, \quad (14)$$

where μ_i is as defined previously. Marroquin [20] uses $w = 1$ with sequential site visitation. Random site visitation with "optimal" w (a value dependent on λ and in the interval $(1, 2)$ - see below under discussion of the GNC algorithm) will also be tried.

2.3 The GNC algorithm

The Graduated Non-convexity (GNC) algorithm is a deterministic procedure for optimising certain non-convex energies associated with piecewise continuous reconstruction problems [3, 5, 8]. It is based on a convex approximation $F^{(1)}$ to the energy F in (4). A family of functions $F^{(p)}$, $p \in [0, 1]$ is defined such that $F^{(1)}$ is convex, $F^{(0)} \equiv F$, and $F^{(p)}$ varies continuously, in a particular prescribed manner, as p decreases from 1 to 0. For $0 \leq p < 1$ the $F^{(p)}$ are non-convex - of the whole family, only $F^{(1)}$ is convex. The $F^{(p)}$ are obtained quite simply by replacing the local interaction energy terms $g(..)$ in (4) by new energy terms $g^{(p)}(..)$ (figure 3). The GNC algorithm for the weak string is given in table 1. Detailed explanation of the algorithm will be found in [8], including such issues as how successive values of p should be chosen and norms for measurement of convergence. The GNC algorithm is distinguished in that it has been proved, for the weak string problem, to converge to the global minimum energy for a significant class of inputs **d** [8]. The proof applies more or less for the practical computer implementation of the algorithm. This is in sharp contradistinction with stochastic algorithms for which only asymptotic results have been obtained [19, 10]. The algorithm used in this paper applies to 1D dense data only. However the algorithm extends very naturally for 2D data, and also (but not quite so naturally) for sparse data.

Because the GNC algorithm is deterministic one might

Choose λ, h_0 (scale and sensitivity).
Set $\alpha = h_0^2\lambda/2$.
SOR parameter: $w = 2/(1+1/\lambda)$.
Function sequence: $p \in \{1, 0.5, 0.25, ..., 1/\lambda\}$.
Nodes: $i \in \{1, ..., N\}$.

Iterate $n = 1, 2, ...$

For $i = 2, ..., N-1$:

$$u_i^{(n+1)} = u_i^{(n)} - \omega\left\{2\left(u_i^{(n)} - d_i\right) + g^{(p)'}\left(u_i^{(n)} - u_{i-1}^{(n+1)}\right) + g^{(p)'}\left(u_i^{(n)} - u_{i+1}^{(n)}\right)\right\} / \left(2 + 4\lambda^2\right)$$

where

$$g^{(p)'}(t) = \begin{cases} 2\lambda^2 t, & \text{if } |t| < q \\ -\frac{1}{2p}(|t| - r)\text{sign}(t), & \text{if } q \le |t| < r \\ 0, & \text{if } |t| \ge r \end{cases}$$

and

$$r^2 = \alpha\left(4p + \frac{1}{\lambda^2}\right), q = \frac{\alpha}{\lambda^2 r}.$$

Initially $p = 1$. Switch to successive p after convergence at current p.
Appropriate modification is necessary at boundaries:

$$u_1^{(n+1)} = u_1^{(n)} - \omega\left\{2\left(u_1^{(n)} - d_1\right) + g^{(p)'}\left(u_1^{(n)} - u_2^{(n)}\right)\right\} / \left(2 + 2\lambda^2\right)$$

and similarly at $i = N$.

Table 1: **The GNC algorithm for the weak string (SOR version).**

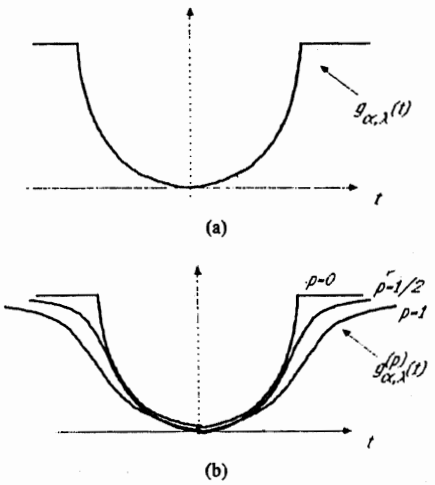


Figure 3: The energy of interaction between neighbouring sites in the weak string computation is governed by the function shown (a). The central part of the function represents a Hooke's law spring, and the outer part a spring pulled past its breaking point. The GNC algorithm replaces this function with a sequence of functions (b).

expect that it should be more efficient than algorithms employing random perturbations. This is precisely what, by controlled experiment, this paper sets out to demonstrate. Now in order to demonstrate the success or failure of an algorithm in reconstruction from a particular set of data $\mathbf{d}$ it is necessary to have some access to the correct solution $\mathbf{u},\mathbf{l}$. For the weak string problem, the solution can be obtained from a dynamic programming algorithm described in the next section.

3 An assay for weak string benchmarks

In this section a new dynamic programming algorithm is presented that delivers an exact solution to the problem of minimising $E(\mathbf{u},\mathbf{l})$. The time-complexity of the algorithm is at worst $O(N^3)$ and can be as little as $O(N^2)$. This compares unfavourably however with GNC whose time-complexity is $O(N\lambda)$ [8]. However, because it is exact the dynamic programming algorithm can be used as an "assay" to verify that a particular reconstruction $(\mathbf{u},\mathbf{l})$ of data $\mathbf{d}$ is indeed optimal (i.e. that it globally minimises $E(\mathbf{u},\mathbf{l})$). Readers wishing to take the assay on trust and look at the results of algorithm evaluation should skip the remainder of this section.

It has already been pointed out [7] that the problem of minimising $F(\mathbf{u})$ can be solved by dynamic programming, provided the real-valued u_i are first quantised into M discrete levels. The time complexity of the resulting algorithm is then $O(NM^2)$ so M must not be too large. However the effect of coarseness of quantisation on accuracy of the solution cannot easily be analysed. Therefore the method is unusable as an assay. Mumford and Shah [22] describe an algorithm for the weak string (or at least for a closely related problem) for which u_i are not required to be quantised. But it relies on an assumption that breakpoints (values of i for which $l_i = 1$) are spaced by a distance that is large compared with the characteristic scale λ. Hence it is not usable in general. As neither of these existing algorithms is suitable for our purpose, a new one is required.

An exact dynamic programming algorithm will be described which requires no quantisation of the u_i. In many cases, moreover, its time-complexity is better than for the earlier dynamic programmming algorithm. The new algorithm will be described in outline here; some further details are given in appendix A.

Given data d_i, $i = 1,..,N$ and choices of parameters α, λ, our problem is to find a global minimum of $E(\mathbf{u},\mathbf{l})$. The first step will be to re-express energy E in terms of the set of breakpoints $\mathbf{L} = \{L_k,\ k = 0,..,K+1\}$ which take values (their positions)

$$L_k \in \{0,..,N-1\},\ 1 \le k \le K, \quad \text{with } L_0 = 0,\ L_{K+1} = N.$$

The constant K is to be chosen in a manner to be described later. Without loss of generality it can be assumed that $L_k \le L_{k+1}$. Given breakpoints, line-variables l_i are

$$l_i = \begin{cases} 1 & \text{if } \exists k \text{ s.t. } L_k = i \\ 0 & \text{otherwise.} \end{cases}$$

The optimal u_i for a given set of break-points could then be obtained simply by minimising $E(\mathbf{u},\mathbf{l})$ which (remember earlier) is convex with respect to $\mathbf{u}$. Hence $\mathbf{u}$ is obtainable from any classical descent algorithm, or else by a recurrence relation with time complexity $O(N)$. In fact we will not be interested in $\mathbf{u}$ - only $\mathbf{l}$ will be required explicitly.

Once energy has been obtained in terms of breakpoints as $E(\mathbf{L})$, it will remain to observe that $E(\mathbf{L})$ is decomposable into a sum of functions each involving only two adjacent breakpoints. Therefore dynamic programming, can be applied in a standard manner, to find the optimal breakpoint set $\mathbf{L}$.

3.1 Expressing energy E in terms of breakpoints

Given a set $\mathbf{L}$ of breakpoints, energy E can be expressed as

$$E(\mathbf{L}) = \sum_{k=0}^{K} \mathcal{E}(L_k, L_{k+1}) + \sum_{k=1}^{K} Z(L_k) \qquad (15)$$

where

$$Z(L_k) = \begin{cases} \alpha & \text{if } 0 < L_k < N \\ 0 & \text{otherwise (i.e. when } L_k = 0) \end{cases} \qquad (16)$$

- the penalty for allowing a discontinuity - and $\mathcal{E}(L_k, L_{k+1})$ is the energy of the continuous length of reconstructed string between breakpoints L_k, L_{k+1} which, from the elastic string model, is:

$$\mathcal{E}(i,j) = \min_{u_{i+1},..,u_j} \left\{ \sum_{m=i+1}^{j} (u_m - d_m)^2 + \lambda^2 \sum_{m=i+1}^{j-1} (u_m - u_{m+1})^2 \right\} \quad \text{for } j \geq i+2 \tag{17}$$

and

$$\mathcal{E}(i,i) = \mathcal{E}(i,i+1) = 0. \tag{18}$$

Note that when $L_k = 0$, $Z(L_k) = 0$ so that node $i = 0$ acts as a "garage" for unrequired breakpoint variables L_k, where they can rest without incurring any breakpoint penalty α. By this means the energy $E(\mathbf{L})$ can represent the energy of any weak string reconstruction with *up to* K breakpoints.

It remains actually to construct $\mathcal{E}(i,j)$ which is a triangular array since it is defined only for $j \geq i$. It is a function of $\mathbf{d}, \lambda, \alpha$ and can be constructed by means of recurrence relations. First the quantity $\mathcal{E}^\dagger(i,j,u_{j+1})$ is defined:

$$\mathcal{E}^\dagger(i,j,u_{j+1}) = \min_{u_{i+1},..,u_j} \left\{ \sum_{m=i+1}^{j+1} (u_m - d_m)^2 + \lambda^2 \sum_{m=i+1}^{j} (u_m - u_{m+1})^2 \right\} \quad \text{for } j > i \tag{19}$$

and

$$\mathcal{E}^\dagger(i,i,u_{i+1}) = (u_{i+1} - d_{i+1})^2. \tag{20}$$

Now from this definition it follows that the following recursive property holds, for $j > i$:

$$\mathcal{E}^\dagger(i,j,u_{j+1}) = \min_{u_j} \left\{ \mathcal{E}^\dagger(i,j-1,u_j) + \lambda^2 (u_j - u_{j+1})^2 + (u_{j+1} - d_{j+1})^2 \right\}. \tag{21}$$

It can now be deduced from (21) and (20), by induction on j, that $\mathcal{E}(i,j,u_{j+1})$ is a quadratic expression in u_{j+1}. From (17), (19) and (20) the quadratic expression is:

$$\mathcal{E}^\dagger(i,j,u_{j+1}) = (u_{j+1} - \overline{u}_{i,j+1})^2 \mathcal{F}_{j+1-i} + \mathcal{E}(i,j+1), \tag{22}$$

where $\mathcal{F}_m$ and $\overline{u}_{i,j}$ are constant coefficients that must be computed and, of course, $\mathcal{E}(i,j)$ is the desired triangular array whose evaluation is the goal of the entire construction above. Substitution of (22) into (21) yields mutual recurrence relations for $\mathcal{F}_m$, $\overline{u}_{i,j}$, $\mathcal{E}(i,j)$ (appendix A.) which enable $\mathcal{E}(i,j)$ to be computed.

Having obtained the triangular array $\mathcal{E}$ we are now in a position to determine K the maximum number of breakpoints. The global minimum energy $E(\mathbf{L})$ is clearly less than the energy in the absence of breakpoints, that is:

$$E(\mathbf{L}) \leq \mathcal{E}(0,N)$$

but if there are K' active breakpoints (breakpoints $L_k \neq 0$ which incur a penalty α) then

$$E(\mathbf{L}) \geq \sum_{k=1}^{K} Z(L_k) = K'\alpha.$$

Hence

$$K' \leq \frac{\mathcal{E}(0,N)}{\alpha}$$

so it is safe to choose

$$K = \left\lfloor \frac{\mathcal{E}(0,N)}{\alpha} \right\rfloor \tag{23}$$

where $\lfloor .. \rfloor$ denotes the integer part of a real-valued quantity.

3.2 Dynamic programming

Now that energy $E(\mathbf{L})$ is in the form of a sum of local functions (15), dynamic programming [1] can be applied. Partial energy functions ϕ_k and policy functions p_k are defined:

$$\phi_0(L_1) = \mathcal{E}(0,L_1), \tag{24}$$

$$\phi_k(L_{k+1}) = \min_{L_k \leq L_{k+1}} \phi_{k-1}(L_k) + \mathcal{E}(L_k, L_{k+1}), \; 1 \leq k \leq K \tag{25}$$

and $p_k(L_{k+1})$ is the value of L_k that minimises (25) above. Construction of all these policy functions (as a set of K N-element tables) has time-complexity $O(KN^2)$. Then the solution for the optimal breakpoints is given by

$$L_{K+1} = N \quad \text{and} \quad L_k = p_k(L_{k+1}) \text{ for } 1 \leq k \leq K. \tag{26}$$

4 Measurements of performance

The previous section described an exact algorithm usable as an assay for weak string benchmarks. In this paper, the bench used for most performance measurements will be an antisymmetric step with $N = 128$ data points:

$$d_i = \begin{cases} 32 & \text{for } 1 \leq i \leq 64 \\ 96 & \text{for } 65 \leq i \leq 128 \end{cases} \tag{27}$$

of height $h = 64$. Varying amounts of uncorrelated, gaussian noise are added as in figure 1. The added noise has standard deviation σ and a relative measure of noise s will frequently be used

$$s = \frac{\sigma}{\sqrt{\alpha}}. \tag{28}$$

It can be shown that, in theory [8], as s falls below $1/\sqrt{2}$, there is a very low probability of "spurious" breakpoints (i.e. other than the "real" one at $i = 64$) occurring in the weak string reconstruction. Moreover, if

$$\alpha < \frac{1}{2}h^2\lambda$$

the real breakpoint *does* appear in the reconstruction (with high probability). Provided $\sigma < 2h$, it is also located precisely correctly (i.e. at $i = 64$). These theoretical predictions have been corroborated by the assay for the data above with added noise of relative amplitude $s = 0.1, 0.2, 0.4$, with $\alpha = 1600$ and a variety of trial values of $\lambda \in [2, 16]$. Numerical details are given in appendix B. Moreover, when $s = 0.8$, the exact, weak string reconstruction contains many spurious breakpoints, also in line with theoretical expectations.

4.1 Measuring rates of convergence

The GNC algorithm, being deterministic, presents little difficulty in measurement of convergence rate. It is simply a matter of running the algorithm repeatedly at gradually increasing precision[1] (and consequently requiring more and more iterations) until breakpoints in the output agree with those in the true reconstruction. Note that only breakpoints, as represented by line-variables $\mathbf{l}$, are required to agree - $\mathbf{u}$ is not tested. This policy is justified by the following observations.

- Determination of breakpoints is the difficult part of the reconstruction problem. Once correct breakpoints are obtained, $\mathbf{u}$ can be calculated by classical relaxation procedures. This is true both of dense data as considered here, and of sparse data [27].
- The u_i are real-valued so they could never be exactly correct, only correct to within some tolerance. Choice of tolerance would be unsatisfactorily arbitrary.

The measure of computation time for the GNC algorithm is then the minimum number of iterations required for a correct output.

Comparisons between algorithms will be in terms of numbers of iterations, ignoring differences in the amount of computation involved in a single iteration which, in any case, are not appreciable.

In the case of stochastic algorithms, convergence rate is much harder to measure because of the random nature of the process. Convergence profiles vary randomly between successive runs of an algorithm. For a given run, a profile is obtained by checking, after each iteration, to see whether $\mathbf{l}$ is in the correct state. Error rate - the proportion of time for which $\mathbf{l}$ is in a state other than the correct one (computed using a time window of 100 iterations) - is plotted as a function of iteration number. Typical examples are shown in figure 4a,b. They are noisy but show a clear trend towards the correct state as the algorithm progresses. The profile in figure 4b was generated from data with 4 times as much noise as in figure 4a. Increased noise in the data has clearly led to a noisier convergence profile. In some cases the profile has been observed eventually to decay to zero error rate, although this is not the case in figure 4a,b.

An error rate threshold of 50% is a natural choice for the following reason. If it is known that the vector $\mathbf{l}^{(n)}$ (the line process at the nth iteration) is in the correct state for more than half of the iterations $n \in \{n_1, .., n_2\}$ then it is sufficient to estimate $\mathbf{l}$ to be:

$$l_i = \begin{cases} 1 & \text{if } \sum_{n=n_1}^{n_2} l_i^{(n)} > n_3 \\ 0 & \text{otherwise} \end{cases} \qquad (29)$$

where n_3 is defined by

$$n_2 = n_1 + 2n_3 - 1.$$

In other words the estimated l_i is simply the value adopted by $l_i^{(n)}$ in the majority of iterations. (This is similar to the 'majority vote" criterion used in defining the class "BPP" of stochastic algorithms.) Convergence could be defined to occur at the largest n for which error rate (measured in a suitable time-window) exceeds 50%. We will adopt a practical lower bound n_L on this value by recording the smallest n for which the error rate falls below 50%. In this respect estimates of convergence rate for stochastic algorithms will be *optimistic*[2]. In the examples of figure 4a,b lower bound n_L is smaller than convergence time as defined above by a factor of 3 or so.

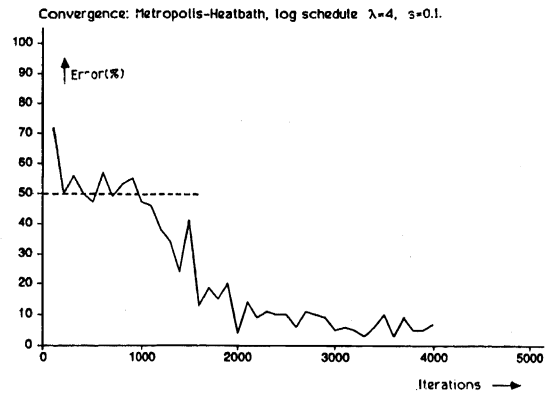
To allow for randomness an average rate $\overline{n}_L$ will be estimated by running the algorithm under test 10 times and computing the average of n_L for those cases in which the algorithm succeeds. (Stochastic algorithms can and do fail by locking out of the correct state - this shows up as a persistent 100% error rate.)

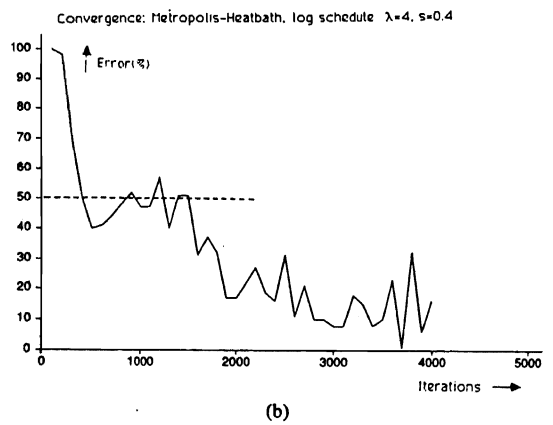
Cooling schedule

An analysis of optimal cooling schedules is outside the scope of this paper, but it is worth noting that a linear schedule is in some ways preferable. Logarithmic and linear schedules for the same data are compared in figure 4b,c. Convergence in the linear case occurs later but

[1] Precision is measured in terms of "absolute norm" [8] which is, in turn, computed from "dynamic norm" - a measure of the change in $\mathbf{u}$ in successive iterations.

[2] Lundy and Mees [19] suggest recording the *first* iteration at which the correct state is hit. This is a still more optimistic measure. But in the first place this is not really applicable to problems like ours that involve real variables. In the second place, even treating $\mathbf{l}$ as the state vector (i.e. ignoring $\mathbf{u}$) so that their measure *can* be applied, results turn out to be qualitatively similar to those obtained by our proposed measure.





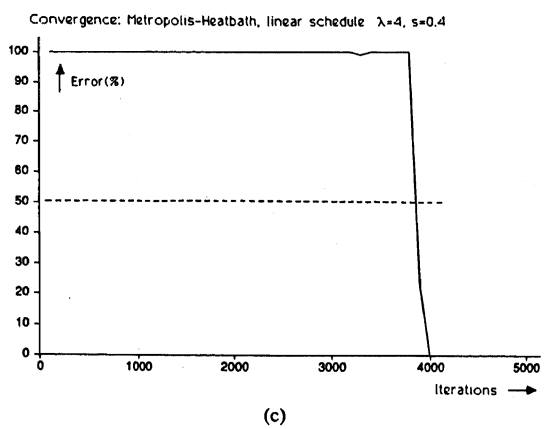


Figure 4: Convergence profiles for individual runs of the Metropolis-Heatbath algorithm. (a) Logarithmic schedule - low noise. (b) Logarithmic schedule - higher noise. (c) Linear schedule - higher noise. Error percentages are averages over 100 successive iterations.

approximately at the same temperature, roughly 10% of the starting temperature. In the logarithmic case, final temperature is strongly determined by initial temperature. Even after 10^6 iterations the final temperature is still about 5% of the initial temperature - see equation (12). In the linear case the final temperature is zero, regardless of starting temperature. Hence performance should be less critically dependent on starting temperature and the following table of results bears this out.

Percentage of successful runs

Starting temp.	Linear	Logarithmic
$T_0 = 2\alpha$	60%	0%
$T_0 = \alpha$	90%	100%
$T_0 = 0.5\alpha$	90%	20%

(Metropolis-Heatbath with $\lambda = 4$, $s = 0.4$, out of 10 runs)

Note also that, in the logarithmic case, it appears that the starting temperature must not be much less than α if the algorithm is to succeed.

In this study a logarithmic schedule with starting temperature $T_0 = \alpha$ will be used throughout.

4.2 Relative performance of stochastic algorithms

A measurement procedure for determination of success and convergence rate of stochastic algorithms has been set up. This makes it possible, first of all, to compare performance of the three stochastic algorithms described in section 2.

Performance is somewhat dependent on the relative noise level s of the data. At low noise (figure 5a) all three algorithms are successful at scales up to $\lambda = 8$. For $\lambda > 8$ the mixed annealing algorithm fails. At a higher noise level (fig 5b) the mixed annealing algorithm fails at all scales. (This continues to be true even when site visitation is random and when the optimal relaxation parameter w is used.) Mixed annealing therefore appears to be much less powerful than the other two. Of those two, Heatbath and Metropolis-Heatbath, it seems from figure 5 that each is of similar power in that they fail (figure 5b) at the same value of λ. But Metropolis-Heatbath is a little more efficient. Therefore Metropolis-Heatbath will be used in comparisons with the GNC algorithm.

4.3 Relative performance of deterministic and stochastic algorithms

At last the main purpose of the paper can be accomplished - comparison of the power of the GNC algorithm

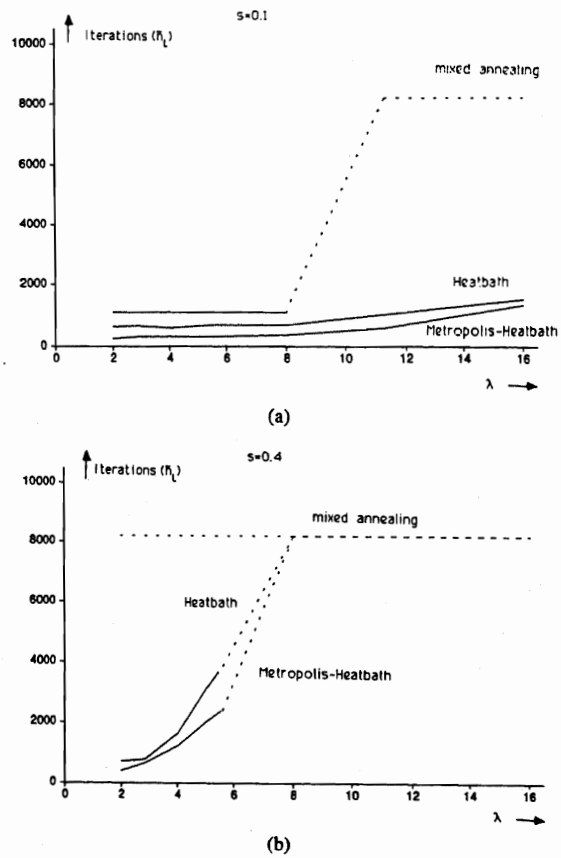


Figure 5: Comparison of stochastic algorithms. (a) Low noise. (b) Higher noise. Dotted lines indicate that algorithm failed on at least some runs.

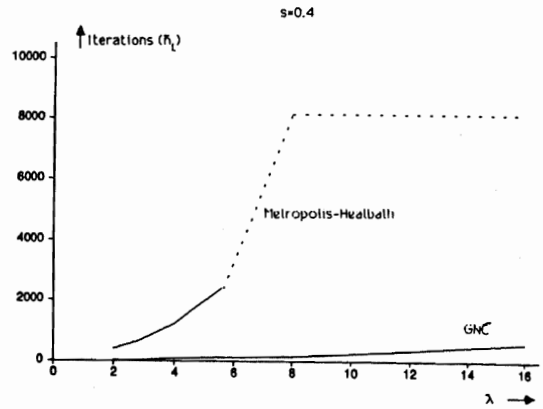


Figure 6: Comparison of stochastic and deterministic algorithms, as a function of varying scale. Dotted lines as figure 5.

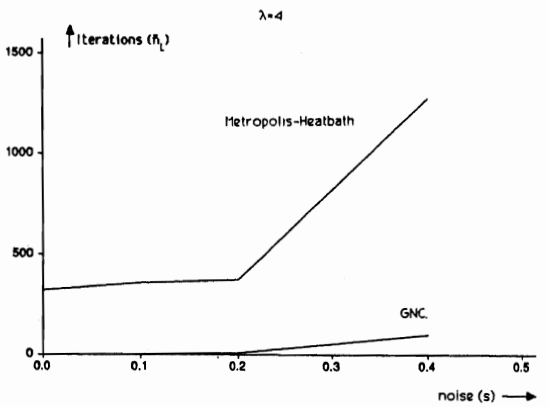


Figure 7: Comparison of stochastic and deterministic algorithms, as a function of varying scale. Dotted lines as figure 5.

with that of the chosen stochastic algorithm. Comparative results are obtained as a function both of scale λ and noise level s. Variation with scale, at moderately high noise level, shows that up to a certain critical scale Metropolis-Heatbath requires between 10 and 20 times more iterations than GNC does (figure 6). Increasing levels of noise (figure 7) and increasing scale (figure 6) cause both algorithms to do more work, though GNC remains comparatively much more efficient. Beyond the critical scale Metropolis-Heatbath fails altogether to find a solution (within 8000 iterations). This justifies the claim, made at the beginning of the paper, that GNC is both more powerful and more efficient. Figure 8 shows that these conclusions hold also for parallel (chequerboard) implementations of the algorithms.

Finally a more complex signal is shown in figure 9. Uncorrelated gaussian noise is added to give a signal-to-noise ratio of about 1:2. The result of the GNC algorithm is shown in figure 9c. Not only is the signal retrieved from

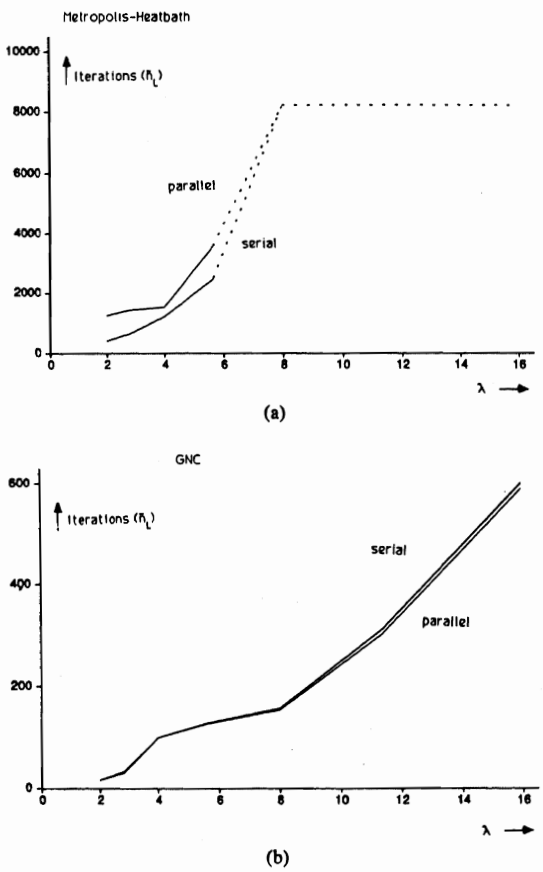


Figure 8: Comparative performance of serial and parallel algorithms at various scales. (a) Stochastic Metropolis-Heatbath. (b) Deterministic GNC.

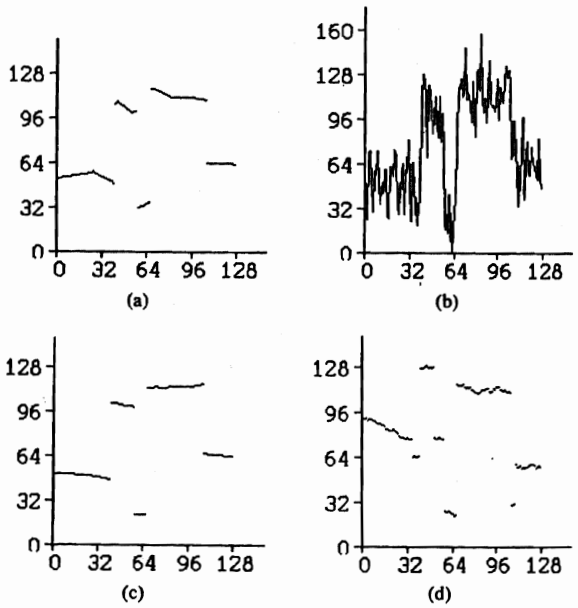


Figure 9: A more complex signal (a) with added noise (b) - signal-to-noise is roughly 1:2. Deterministic GNC algorithm produces a correct reconstruction (c) with parameters $\lambda = 16, \alpha = 5000$. Stochastic algorithm fails (d).

the noise, but the reconstruction is verified by the assay. A reasonably large scale λ must be used to retrieve the signal because of the high noise level (s=0.23). (This is because high resistance to noise (large α), with reasonable sensitivity (small $\sqrt{\alpha/\lambda}$), demands large scale λ.) The stochastic algorithm can be expected to fail. This is just what it does in 100% of 10 runs. In a typical run it never visits the correct state[3]. An example state (after 8000 iterations) contains extra, spurious breakpoints (figure 9d) in addition to some correct ones.

5 Conclusions

First of all, a "test-bed" for a piecewise continuous reconstruction problem has been established, by means of the assay described in section 3. The dynamic programming algorithm is relatively straightforward to implement[4]. Reconstructions for data used in this paper have been verified by the assay for certain values of α, λ, s (appendix B.).

Comparison of the deterministic GNC algorithm with three stochastic algorithms has shown the latter to be considerably less efficient. Furthermore they are less powerful in that they cannot practically deliver correct solutions for problems involving moderately high levels of noise (and which therefore demand large scale in recon-

[3]It therefore fails even under the Lundy and Mees [19] criterion
[4]Code in POP11 is available from the author on request.

struction). This is true both of serial and parallel implementations of the algorithms.

Finally the theoretical power of simulated annealing, at least for the practical annealing schedules used, is not realised in practice. This is consistent with experimental results from another study [12]. This suggests that the apparent freedom to "design" MRFs to represent prior knowledge is severely curtailed in practice, since it is unknown whether available estimation techniques will be powerful enough to apply that knowledge. Hence this paper reinforces earlier arguments [8] that the potential of MRFs as a *general* vehicle for specifying and integrating visual tasks must be regarded as highly questionable.

Acknowledgments

This work was supported by SERC grant GR/D 1439.6, by the University of Edinburgh and by the Royal Society of London's IBM Research Fellowship. I am very grateful to Andrew Zisserman, Bernard Buxton, Alex Kashko and Alistair Sinclair for helpful comments.

References

[1] Bellman,R. and Dreyfus,S. (1962). *Applied dynamic programming.* Princeton Univ. Press, Princeton, U.S..

[2] Besag,J. (1986). On the statistical analysis of dirty pictures. *J. Roy. Stat. Soc. Lond. B*, 48, 259-302.

[3] Blake,A. (1983). The least disturbance principle and weak constraints. *Pattern Recognition Letters*, 1, 393-399.

[4] Blake,A. (1985). Boundary conditions for lightness computation in Mondrian World. *Computer vision graphics and image processing*, 32, 314-327.

[5] Blake,A. and Zisserman,A. (1986). Some properties of weak continuity constraints and the GNC algorithm. *Proc. CVPR Miami*, 656-661.

[6] Blake,A. and Zisserman,A. (1986). Invariant surface reconstruction using weak continuity constraints. *Proc. CVPR Miami*, 62-67.

[7] Blake,A., Zisserman,A. and Papoulias,A.V. (1986). Weak continuity constraints generate uniform scale-space descriptions of plane curves. *Proc ECAI, Brighton, 1986*, 518-528.

[8] Blake,A. and Zisserman,A. (1987). *Visual Reconstruction.* MIT Press, Cambridge, USA.

[9] Derin,H. and Elliot,H. (1987). Modelling and segmentation of noisy and textured images using Gibbs Random Fields. *IEEE PAMI*, 9, 1, 39-55.

[10] Geman,S. and Geman,D. (1984). Stochastic Relaxation, Gibbs distribution, and Bayesian restoration of images. *IEEE PAMI*, 6, 721-741.

[11] Grimson,W.E.L. (1981). *From images to surfaces.* MIT Press, Cambridge, USA.

[12] Greig,D.M., Porteous,B.T. and Seheult,A.H. (1986). Discussion of Besag's paper. *J. Roy. Stat. Soc. Lond. B*, 48, 282-284.

[13] Horn,B.K.P. (1974). Determining lightness from an image. *Computer Graphics and Image Processing*, 3, 277-299.

[14] Horn,B.K.P and Schunk,B.G. (1981). Determining optical flow. *Computer Vision*, ed. Brady,J.M.

[15] Ikeuchi,K. and Horn,B.K.P. (1981), Numerical shape from shading and occluding boundaries, *Computer Vision*, ed. Brady,J.M., 141-184.

[16] Kashko,A. (1987). A parallel approach to Graduated Nonconvexity on a SIMD machine. Report, Dept. Computer Science, Queen Mary College, London.

[17] Kirpatrick,S., Gellatt,C.D. and Vecchi,M.P. (1982). Optimisation by simulated annealing. IBM Thomas J. Watson Research Centre, Yorktown Heights, NY, USA.

[18] Koch,C., Marroquin,J.L. and Yuille,A. (1986). Analog networks in early vision. *Proc. Nat. Acad. Sci. USA*, 83, 4263-4267.

[19] Lundy,M. and Mees,A. (1984). Convergence of the annealing algorithm. Report, DAMTP, University of Cambridge.

[20] Marroquin,J. (1984). Surface reconstruction preserving discontinuities. Memo 792, AI Laboratory, MIT, Cambridge, USA.

[21] Metropolis,N., Rosenbluth,A.W., Rosenbluth,M.N., Teller,A.H. and Teller,E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 6, 1087.

[22] Mumford,D. and Shah,J. (1985). Boundary detection by minimising functionals. *Proc. IEEE CVPR conf.*, 22.

[23] Murray,D.W. and Buxton,B. (1986). Scene segmentation from visual motion using global optimisation. *IEEE PAMI*, 9, 2, 220-228.

[24] Nagel,H.H. (1983). Constraints for the estimation of displacement vector fields from image sequences. *Proc. IJCAI Karlsruhe*, 945-951.

[25] Smith,W.E., Barrett,H.H. and Paxman,R.G. (1983). Reconstruction of objects from coded images by simulated annealing. *Optics letters*, 8, 4, 199-201.

[26] Strang,G. and Fix,G.J. (1973). *An analysis of the finite element method.* Prentice-Hall, Englewood Cliffs, USA.