J Indian Inst Sci., Jan.–Feb 1998, 78, 31–43
© Indian Institute of Science

Edge detection through a time-homogeneous markov chain

UMA S. RANJAN, V. S. BORKAR* AND P. S. SASTRY
Department of Electrical Engineering, Indian Institute of Science, Bangalore 560 012, India.
*Department of Computer Science and Automation, Research supported by Grant No. 26/1/92-G from DAE, Govt. of India.

Abstract

A Monte Carlo-type stochastic algorithm for edge detection is presented. It takes a suitable monotone decreasing function of the norm of gradient of a low-pass filtered image as 'cost' and seeks the 'valleys' of the associated landscape Simulation studies and mathematical analysis are presented.

1. Introduction

Computational Vision is often classified into two stages – low level vision and high-level vision. Low-level vision techniques abstract features or properties from a digital image that may be used as an input to a higher-end system. The main requirements of low-level visual processing is that it be uniformly applicable to as large a class of images as possible. One of the important tasks of a low-level visual processing system is edge detection.

Ideally an edge detector should extract features that facilitate detection of object boundaries and figure-ground separation. Design of all edge detectors is based on the fact that object boundaries show up as sharp changes in the 2D intensity function represented by the image. Most edge detectors attempt to find object boundaries through detection of maxima in the gradient of the intensity function (or detection of zero-crossings in the second derivative). However, intensity changes in an image may occur due to a variety of other causes such as shadows, noise, texture *etc* thus giving rise to spurious boundaries. One way to avoid spurious edges is to impose additional constraints, corresponding to regularization. Most techniques of regularization involve either approximating the intensity function by polynomials, splines or other piecewise smooth functions[6,3] or optimising an energy functional consisting of suitable regularizing terms[8,4,5] in addition to gradient of the intensity. The latter method is attractive because it allows an ordering among solutions and several constraints may be imposed simultaneously. Unfortunately, the energy functionals turn out to be usually highly non-convex and require computationally expensive methods such as Simulated Annealing for good performance.

In this paper, we propose an algorithm which has far less computational complexity in comparison to stochastic techniques such as Simulated Annealing. This has been achieved through defining a pixel-based cost rather than a configuration-based cost. The solution is then not a single point in the configuration space but a set of points on the pixel space which have a relatively low value of the cost function. This solution set is tracked by means of a dynamical system moving over the pixel array. This approach combines both local and global characteristics.

The local nature of computations reduces computational complexity while the global tracking provides immunity against noise.

We model edges as points corresponding to a relatively high value of $|\nabla f|^2$ where f is the image function $I(x, y)$ filtered with a 2-D Gaussian mask $G(x,y)$. Filtering is done to partially compensate for noise-induced rapid spatial variation of intensities, though this also results in the loss of some edges. Edge points, however, need not correspond either to local or global maxima of $|\nabla f|^2$, since there may be slight variations in the value of $|\nabla f|^2$ along the edge, though they are still relatively high w.r.t. points off the edge. Hence this is not a conventional optimisation problem and standard gradient techniques cannot be applied.

The algorithm presented here tracks edge points by means of a stochastic process which spends a relatively large fraction of time at points corresponding to higher values of $|\Delta f|^2$. We also propose an equivalent parallel algorithm which overcomes some of the limitations of the sequential algorithm.

2. The sequential algorithm

A time-homogeneous Markov chain $X(t)$, $t \geq 0$ is defined on the state space of pixel array $S = \{(i, j) : 1 \leq i \leq N, 1 \leq j \leq N\}$. A neighbourhood structure $N(.)$ is defined on the state space of pixel array S such that the set of neighbours $N(i, j)$ of pixel (i, j) is the 3×3 neighbourhood $\{(m, n) | ((m - i)^2 + (n - j)^2))^{1/2} \leq 2, (m, n) \neq (i, j)\}$

Thus a typical neighbourhood looks as in Figure 1 where we have assumed that the pixel lattice is embedded on a torus in such a way that the top and bottom edges as well as the right and left edges are adjacent, while preserving the orientations.

$g : S \times S \to [0,1]$ is a selection probability function satisfying

1. $\Sigma_{(m, n) \in S}\, g[(i, j), (m, n)] = 1$
2. $g[(i,j),(m,n)] \begin{cases} > 0 & \text{if}(m,n) \in N(i,j) \\ = 0 & \text{otherwise} \end{cases}$

$g[.,.]$ is constrained to be symmetric, *i.e.*, $g[(i, j), (m, n)] = g[(m, n), (i, j)]$.

The one-step transition probability $p[.,.]$ of the Markov process $X(t)$ is given by

$$\begin{aligned} p[(i, j), (m, n)] &= p\{X(t + 1) = (m, n) \mid X(t) = (i, j)\} \\ &= g[(i, j), (m, n)] \exp\{-(c(m, n) - c(i, j))^+ /T\} \\ p[(i, j), (i, j)] &= 1 - \Sigma_{(m, n) \in N(i,j)}\, p[(i, j), (m, n)] \end{aligned}$$

```
*   *   *

*   •   *

*   *   *
```

FIG. 1. • : point under consideration *: neighbouring points.

Here $\exp\{-(c(m, n) - c(i, j))^{+}/T\}$ is the probability that the neighbouring state (m, n) will be accepted conditioned on its selection and $c(i, j)$ is the cost associated with state (i, j) where $c(i, j) = -|\nabla f|^2(i, j)$. Thus, points corresponding to relatively high values of the gradient function correspond to relatively low cost values. It is clear from the graph of the Markov chain that for every pair of states (i, j), $(p, q) \in S$, $\exists (i_0, j_0), (i_1, j_1), \ldots, (i_n, j_n) \in S$, $(i_0, j_0) = (i, j)$; $(i_n, j_n) = (p, q)$ such that $p[(i_k, j_k), (i_{k+1}, j_{k+1})] > 0$, $k = 0, 1, \ldots, n-1$. Hence the chain consists of a single communicating class. Moreover, there is at least one state $(i, j) \in S$ (e.g. the point of minimum value of the cost function) such that $p[(i, j), (i, j)] > 0$. Hence the chain is aperiodic.

$X(t)$ is thus an aperiodic, irreducible Markov chain. The unique stationary distribution $\pi(.)$ where

$$\pi(i) = \lim_{t \to \infty} \Pr\{X(t) = i\}$$

of such a chain exists[1] and can be obtained from the global balance equation

$$\sum_{(i,j)} \pi(i,j) p\left[(i,j),(m,n)\right] = \pi(m,n) \tag{1}$$

A sufficient condition for (1) is the detailed balance equation which implies the global balance equation. From the detailed balance

$$\pi(i, j)\, p[(i, j), (m, n)] = \pi(m, n)\, p[(m, n), (i, j)] \tag{2}$$

and the assumption that $g[.,.]$ is symmetric,

$$\pi(i,j) = \frac{e^{-c(i,j)/T}}{Z(T)} \tag{3}$$

where

$$Z(T) = \sum_{(m,n) \in S} e^{-c(m,n)/T}$$

It can be seen that at a given temperature T,

$$\frac{\pi(i,j)}{\pi(m,n)} = e^{-\{c(i,j)-c(m,n)\}/T} > 1$$

if $c(i, j) < c(m, n)$. Since the limiting distribution $\pi(.)$ also gives the long-run mean fraction of time that the process $X(t)$ spends in a state,

$$\pi(i,j) \approx \frac{1}{T} \sum_{t=0}^{T} I\{X(t) = (i,j)\}$$

for large T. Thus, a higher value of $\pi(.)$ for a state implies that the state is visited more often than other states by the process.

Thus the algorithm sequentially "tracks" the edges independent of the starting point.

2.1. *Effect of Temperature*

Let T_1, T_2 be any two temperatures, $T_1 > T_2$. The stationary distribution at T_1 and T_2 are given by

$$\pi(i,j)|T_1 = \frac{e^{-c(i,j)/T_1}}{Z(T_1)} \tag{4}$$

and

$$\pi(i,j)|T_2 = \frac{e^{-c(i,j)/T_2}}{Z(T_2)} \tag{5}$$

respectively, where $Z(T_i)$ is the normalising factor. Let (i, j), (m, n) be such that $c(i, j) < c(m, n)$. Then

$$\begin{aligned}\frac{\pi(i,j)}{\pi(m,n)}|T_1 &= e^{-\{c(i,j)-c(m,n)\}/T_1}\\ &< e^{-\{c(i,j)-c(m,n)\}/T_2}\\ &= \frac{\pi(i,j)}{\pi(m,n)}|T_2\end{aligned}$$

Thus, at lower temperatures, the stationary distribution has relatively higher peaks at points of lower cost. Hence the process will spend less time at points of relatively high cost (≈ non-edge points) at lower temperatures. However, since the probability of an uphill transition being accepted is low at lower temperatures, the chain will require a longer time to converge to its equilibrium behaviour.

At higher temperatures, the chain moves faster over the state space, but the relative difference between the probability assigned by the invariant measures to points of higher and lower costs (≈ non-edge points and edge points) is lower. Hence a trade-off between the two is required.

2.2. *Determination of Threshold*

It has been shown that if $c(i) < c(j)$ and $T_1 > T_2$,

$$\frac{\pi(i)}{\pi(j)}|T_1 < \frac{\pi(i)}{\pi(j)}|T_2$$

At $T = \infty$, $\pi(i) = \pi(j) = \frac{1}{N^2}\,\forall i, j.$. For any finite $T < \infty$, points of "relatively lower cost" will have a value of $\pi(.) \geq \frac{1}{N^2}$ and points of "relatively higher cost" a value of $\pi(.) \leq \frac{1}{N^2}$ in order to maintain a total probability of 1. Thus $\frac{1}{N^2}$ is a reasonable threshold for identification of points of "relatively low cost". In terms of the count at a point, the threshold equals [$\frac{1}{N^2}$ length of the run of the Markov chain].

2.3. *Discussion*

In order to enhance the performance of the edge detector, some modifications were made:

1. The cost as defined is unbounded and hence,results in thick edges[2]. Any linear scaling of the cost function is equivalent to a mere change of temperature. However, a non-linear scaling of the cost function which emphasizes large differences of cost while suppressing small differences smooths out smaller local minima in the vicinity of the larger minima. The function $|\nabla f|^2$ was linearly scaled between 0 and a value close to $\pi'/2$ and its tangent was taken as the cost function. This value will henceforth be referred to as cost[.].
2. g[.,.] was formulated such that the neighbours were not chosen with equal probability, but neighbours of lower cost were given a higher probability of selection. Maximum probability was assigned to the direction of minimum cost. The minimum probability was assigned to the pixel in the direction opposite to this, and intermediate values were assigned to other directions. The actual values chosen were as in Figure 2 where (p, q) is such that $cost(p, q) = min_{(m,\, n) \in N(i,\, j)}\ cost(m, n)$. Moreover, the neighbourhood structure was not embedded on a torus, but on a rectangular grid so that the cardinality of the neighbourhood is 3 for a corner point and 5 for a non-corner edge point. In such a case, the selection probabilities of the points deleted from the neighbourhood are added to that of the (edge or corner) point (m,n), which is therefore selected with a non-zero probability. This violates the symmetry condition of g[.] imposed for theoretical analysis; however, this preferential selection of neighbours aids movement along the edge rather than either movement away from it or a slow diffusion along it.
3. A limit on the count at a point is set and if this count is exceeded, the point is marked as an inhibited point. If the process goes to an inhibited point, it is perturbed to one of its neighbours. This is done so that the process does not detect only a part of an edge due to the presence of a non-zero gradient along the edge. The count at which points are inhibited is determined as a percentage of the total number of iterations.
4. Since the starting point is chosen arbitrarily, this point might well have been chosen very far from an edge contour and a large number of iterations are required to reach the edge contour. Also, only the edges in the vicinity of the starting point may be detected in a single run of the algorithm. This can be rectified by starting the process simultaneously

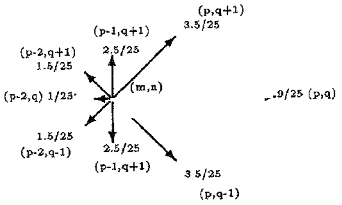

FIG. 2. Selection probability diagram.

at more than one point. This issue has been discussed again in the parallel algorithm in the next section.

3. A parallel algorithm

A better performance than that of the sequential algorithm is achieved by an asymptotically equivalent parallel algorithm wherein we start one such chain at each pixel. Two or more chains coalesce when they meet (*i.e.*, when they make transition into the same pixel at some instant) and from then on move together.

We describe this algorithm by adopting the following notation.

Let $M = N^2$ where $N^2 = |S|$ (= number of pixels) and let $S_p = S^M$ ($= S \times S \times ... \times S$, *M times*). Thus, a typical point in S_p is $x = [x_1, x_2, ..., x_M]$, $x_i \in S$ for all i. Let $\{p_1, p_2, ..., p_M\}$ be an enumeration of pixels. Let $X(t) = [X_1(t), X_2(t), ..., X_M(t)]$, $t = 0, 1, ...$ be an S_p-valued process described as follows:

$X_i(0) = p_i$ for $1 \le i \le M$. (Thus, the i^{th} component of $X(.)$ is an S-valued process starting at p_i). At each $t \ge 0$, the following occurs.

At each pixel p_i, a neighbour p_j is selected from $N(p_i)$ according to prescribed selection probabilities.(We use the uniform probability for this selection.This satisfies the detailed balance at all points except the boundary points. We neglect the boundary effects caused by the imbalance.)

Next, p_j is "accepted" with probability $\exp\{-(c(p_j) - c(p_i))^+/T\}$. If p_j is "accepted", the processes $X_k(t)$, $1 \le k \le M$ such that $X_k(t) = p_i$, move to p_j at time $t+1$. If not, they remain at p_i. The selection and acceptance at distinct pixels is performed in a statistically independent manner.Thus, for a fixed i, $\{X_i(t)\}$ is a copy of the process defining the serial version described earlier, starting at p_i, and therefore exhibits the same asymptotic behaviour. $\{X_i(.)\}$, however,are not independent. In fact, they are highly correlated as will become clear later. The following theorem suggests that the intuition behind the serial version which justifies its use for edge-detection, carries over *in toto* to the parallel version.

Let $N_i(t) = \Sigma_{k=1}^{M} I\{X_k(t) = p_i\}, 1 \le i \le M$. Thus, $N_i(t)$ = number of processes at pixel p_i at time t. Clearly, $N_i(0) = 1$ for all i.

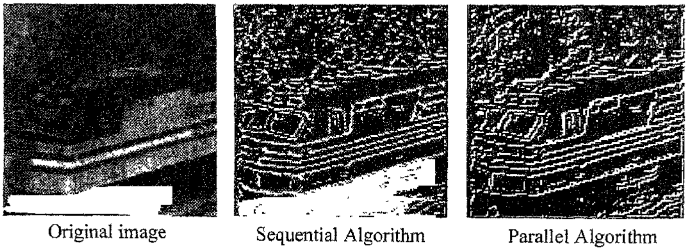

FIG. 3. Train image.

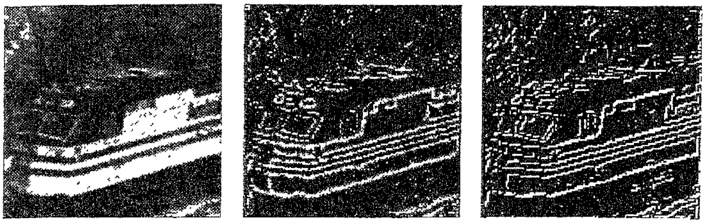

FIG 4. Train image corrupted with noise ($\sigma = 20$).

Theorem 1: $\alpha_i = \lim_{t\to\infty} E[N_i/H]$ *exists for all i and is proportional to* $\pi(p_i)$.

Proof

$$\begin{aligned}
&\lim_{t\to\infty} E[N_i(t)]\\
&= \lim_{t\to\infty} E\left[\sum_{k=1}^{M} I\{X_k(t) = p_i\}\right]\\
&= \lim_{t\to\infty} \sum_{k=1}^{M} p\{X_k(t) = p_i\}\\
&= \sum_{k=1}^{M} \lim_{t\to\infty} p\{X_k(t) = p_i\}\\
&= \sum_{k=1}^{M} \pi(p_i)
\end{aligned}$$

since each $X_k(t)$ has the same asymptotic behaviour as in the serial version.

$$\alpha_i = M\pi(p_i),$$

i.e., α_i is proportional to $\pi(p_i)$

The following theorem shows that the serial and the parallel versions are asymptotically equivalent.

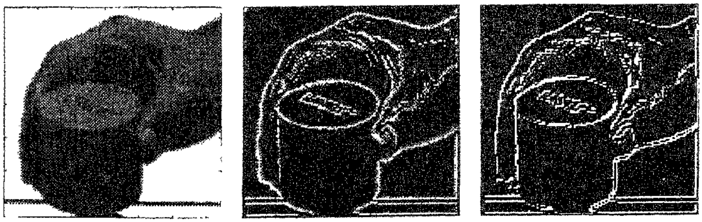

FIG 5. Unix image.

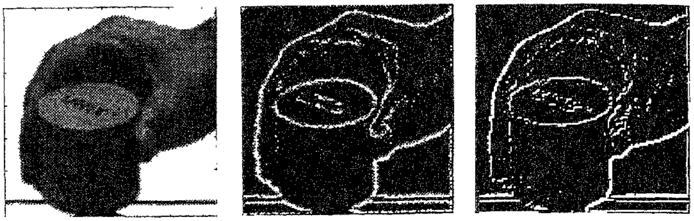

FIG. 6. Unix image corrupted with noise ($\sigma = 20$)

Let $S_p^* \subset S_p$ be the set $\{x = [x_1, \ldots, x_M] \in S_p \mid x_j = x_1 \forall j\}$

Theorem 2: 1. S_p^* *is an aperiodic communicating class of states.*

2. *All states in* $S_p \backslash S_p^*$ *are transient.*

Proof

1. From the explicit construction, it is clear that if $X_j(t_0) = X_1(t_0)$ at some random t_0 depending on j, it remains so for all $t \geq t_0$. Thus S_p^* is closed. Once in S_p^*, all components of $X(.)$ move together as a block and exhibit the same dynamics as that of the serial version. Hence 1 follows.
2. For any $x \in S_p \backslash S_p^*$, it is clear that the probability of $X(.)$ hitting S_p^* in finite time after t, conditioned on $X(t) = x$, is strictly positive. Hence the second claim.

Remark 1: *Theorem 2 automatically implies Theorem 1. We have established the latter separately because the convergence of distributions therein seems to occur much faster (i.e., within a typical run of the algorithm) than the coalescing of processes implicit in the former. In fact,*

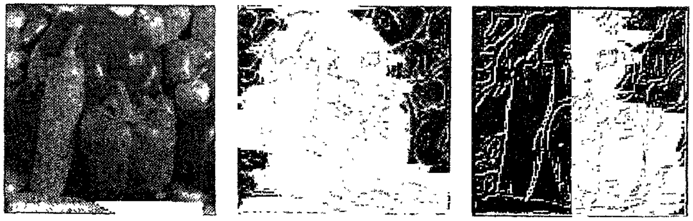

FIG. 7. Peppers image.

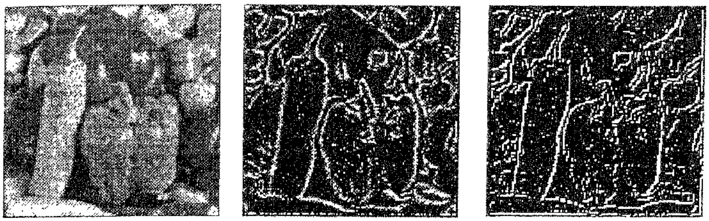

FIG. 8 Peppers image corrupted with noise ($\sigma = 20$).

given that the parallel version employs simple thresholding for identifying edge points, it is not desirable to wait till all processes coalesce.

3.1. *Determination of Threshold*

Following the same argument as in the sequential version, points of "relatively higher cost" have an expected value of count $\alpha(i) < N^2\frac{1}{N^2}$. Hence, the threshold on alpha is $\lfloor \alpha(i) \rfloor = 0$.

3.2. *Discussion*

The parallel algorithm has the following advantages over the sequential algorithm:

1. No ad hoc modifications are required to take care of initial conditions or non-zero gradient along the edge (since the parallel algorithm starts a sequential Markov chain at every pixel).
2. No explicit count is required to be maintained w.r.t. time.
3. The threshold on the number of processes has been set to 0. This simplifies the thresholding operation in the neural network implementation, to be discussed in the next section.

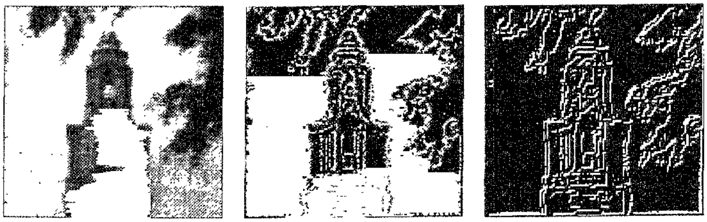

FIG. 9. Institute image.

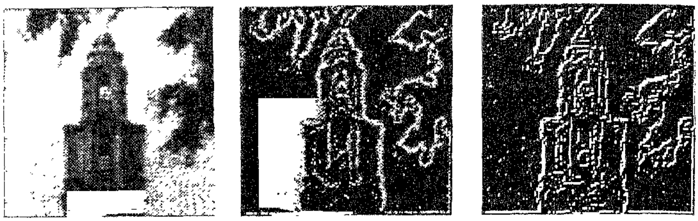

FIG. 10. Institute image corrupted with noise ($\sigma = 20$).

4. A Neural Network Implementation

The dynamic process described in the earlier sections has been cast here in the framework of a neural network. The network consists of three layers - an input layer $u(.)$ an output layer $w(.)$ and an intermediate hidden layer $v(.)$. Each layer consists of an array of $N \times N$ nodes.

The input layer contains the image data, or equivalently, the cost function values. The values of the nodes are given by

$$u(i, j) = -c(i, j). \tag{6}$$

The intermediate layer contains the values of the number of processes $N_i(t)$ at site i as defined in the parallel algorithm.

$$v^{(t)}(i, j) = N_{(i,j)}(t); \tag{7}$$

Each node (i, j) in the intermediate layer is connected to all nodes in its 3×3 neighbourhood so that the configuration of the network evolves as a result of local interactions. The nodes of the intermediate layer are updated in the following manner:

The value $v^{(t+1)}(i, j)$ of node (i, j) at instant $(t + 1)$ is given by

$$\begin{aligned} v^{(t+1)}(i,j) = v^{(t)}(i,j) + &\sum_{(m,n)\in N(i,j)} I\{(m,n) \rightarrow (i,j)\} v^{(t)}(m,n) \\ - v^{(t)}(i,j) &\sum_{(m,n)\in N(i,j)} I\{(i,j) \rightarrow (m,n)\} \end{aligned} \tag{8}$$

where $I\{(m, n) \rightarrow (i, j)\}$ is the indicator of the event that a transition has occurred from node (m, n) to node (i, j) at the instant of evaluation, *i.e.*, $I\{(m, n) \rightarrow (i, j)\}$ is 1 if this transition occurs and 0 otherwise.

The second term on the R.H.S. of Equation 8 corresponds to the processes that enter the node in consideration and the third term corresponds to the event that the processes currently at the node leave the node.

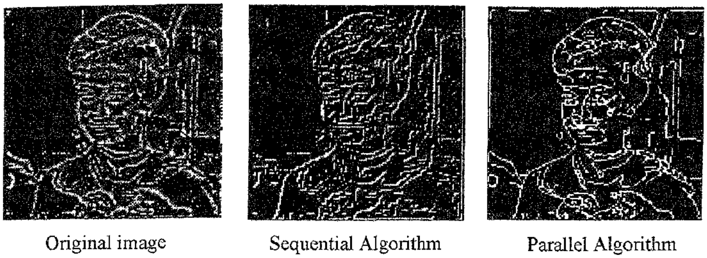

FIG. 11. Results on IEEE girl image.

$v^{(t+1)}(i,j)$ is thus a random variable with conditional Expectation

$$E\left[v^{(t+1)}(i,j)|v^{(t)}(i,j);v^{(t)}(m,n),(m,n)\in N(i,j)\right]=$$
$$v^{(t)}(i,j)+\Sigma_{(m,n)\in N(i,j)}v^{(t)}(m,n)p[(m,n),(i,j)]$$
$$-v^{(t)}(i,j)\Sigma_{(m,n)\in N(i,j)}p[(i,j),(m,n)] \tag{9}$$

where $p[.,.]$ is the transition probability as defined earlier.

By Theorem 1, the expected values of $v^{(t)}(i,j)$ converge to a value proportional to the invariant measure of (i,j).

The output layer contains the edge image, thresholded as before.

$$w(i,j)=\begin{cases}1 & \text{if } v(i,j)>0\\ 0 & \text{otherwise}\end{cases}$$

5. Results

The images were filtered initially with a 5×5 zero-mean Gaussian mask of variance $\sigma^2=1.0$. The parameters chosen for the algorithms on 128×128 images were as follows:

- Sequential Algorithm:
  1. Temperature = 0.25
  2. Length of the run of the homogeneous Markov chain = 2 million iterations
  3. Threshold = 150.
- Parallel Algorithm :
  1. Temperature = 0.25
  2. Number of passes = Length of the homogeneous chain/ Number of pixels(image size) ≈ 150.
  3. Threshold = 0.

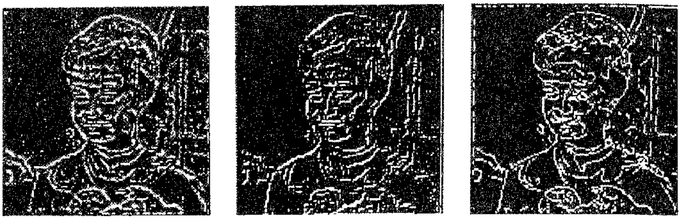

FIG 12 Girl image corrupted with noise of $\sigma = 10$.

We note that the results obtained by the stochastic algorithm are comparable to those reported in literature[7,4] using simulated annealing under conditions of no noise. Moreover, no false edges are reported on smoothly shaded surfaces. This is due to the fact that smoothly shaded areas correspond to relatively smaller local minima which are not detected by our algorithm.

In terms of time taken, the typical time taken for a 128×128 image is approximately 6 minutes of CPU time on a CD-IRIS workstation. This drastic reduction of time is due to the fact that the cardinality of the solution space is much less in case of the algorithms presented here (from $2^{|S|}$ for simulated annealing to $|S|$ for our stochastic algorithm, where S is the number of pixels in the image array).

The parallel algorithm offers a further reduction in computational time. Moreover, since each pass of the algorithm operates only on pixels having a non-zero value, even a sequential implementation of the parallel algorithm is much faster (typically 2-3 minutes on the same computing platform) than the sequential algorithm.

The algorithms were also run on 8-bit images corrupted with white Gaussian noise. The strength of these algorithms is that they are robust and perform well even in the presence of noise. The algorithms were run on 2-D images corrupted with white Gaussian noise of standard deviation $\sigma = 10$. The values of the parameters chosen were the same as before. Figure 3–10 show the results of both the sequential and the parallel algorithm on various real-world images. In Figure 11 and Figure 12, the results of the sequential and parallel algorithms are presented along with a comparison by the Canny edge detector. The implementation of the Canny detector used was a matlab implementation by Prof. Perona of Caltech.

References

1. BHAT, R. R., *Modern Probability Theory*, Wiley Eastern Ltd., 2nd edition, 1985.
2. GRIFF L. BILBRO, WESLEY E. SYNDER, STEPHEN J. GARNIER AND JAMES W. GAULT, Mean Field Annealing: A formalism for contructing GNC-Like algorithms. *IEEE Trans. Neural Networks*, 1992, 3(1), 131–139.
3. ANDREW BLAKE AND ANDREW ZISSERMAN, *Visual Reconstruction*, MIT Press, Cambridge, MA, 1987.

4. GEMAN, D. Stochastic model for boundary detection. *Image and vision computing*, 1987, **5(2)**, 61–65.

5. GEMAN, D., GEMAN, S., GRAFFIGNE, C. AND DORY, P., Boundary detection by constrained optimization, *IEEE Trans. Pattern Analysis and machine intelligence*, 1990, pp. 609–628.

6. ROBERT M. HARALICK, Digital step edges from zero-crossing of second directional derivatives, *IEEE Trans. pattern analysis and machine intelligence*, 1984, **6**(1), 58–68.

7. TAN, H. L., GELFAND, S. B. AND DELP, E. J., A cost minimization approach to edge detection using simulated annealing, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1991, **14(1)**, 3–18.

8. TORRE AND POGGIO, T., On edge detection, Technical report AI memo 768, Massachusetts Institute of Technology, August 1984.