IMAGE ENHANCEMENT USING VERY FAST SIMULATED REANNEALING

Balaji Natrajan and Bruce E. Rosen

Division of Computer Science
The University of Texas at San Antonio
San Antonio, TX, 78249
bnatraja,rosen@ringer.cs.utsa.edu

ABSTRACT

We compare the Random Greedy algorithm, the Cauchy Annealing algorithm and the Very Fast Simulated Reannealing algorithm (VFSR) on a piecewise constant image enhancement task. We tested these algorithms on a noise corrupted 64x64 and 256x256 image and show that the VFSR algorithm converges to better solutions than Cauchy annealing, or the random greedy search algorithm.

I. INTRODUCTION

The goal of image enhancement is to improve the quality of an image subject to various constraints. If image quality can be quantified and objectively measured, a cost function reflecting the quality of a solution (image) may be constructed allowing the quality of different solutions to be compared. The ideal cost function is one which, by definition, has a global minimum corresponding to the most "enhanced" image. Since the cost function is usually combinatorial, stochastic approximation methods such as simulated annealing (SA) can be used to find near optimal enhanced images. Simulated annealing is a stochastic neighborhood search algorithm that simulates the "annealing" process using the Metropolis algorithm [2]. The term *annealing* is derived from an analogy with the metallurgical process for refining a metal in a heat bath to its lowest energy state through slow cooling [10].

The Metropolis algorithm can be applied to image enhancement problems. Given a current solution image i with cost $E(i)$, then a subsequent neighboring solution j is generated by applying a perturbation mechanism that transforms the current solution by a small distortion, e.g., by changing the intensity of single pixel in the current solution. The cost of this next solution is $E(j)$. If the energy difference between the current solution and the last solution, $E(j) - E(i)$, is less than or equal to 0, the state j is accepted as the current solution. If the energy difference is greater than 0, the solution j is accepted with a certain probability which is given by

$$exp(\frac{E(i) - E(j)}{T}),$$

where T denotes the current value of the control parameter called the temperature. This acceptance rule is known as the *Metropolis criterion* and the algorithm is known as the *Metropolis algorithm* [2].

A. Simulated Annealing

The simulated annealing algorithm is basically iterations of the Metropolis algorithm, evaluated at decreasing values of the control parameter, the temperature. This technique was first popularized by [12] to solve combinatorial optimization problems. Using Markov chain theory, simulated annealing can be shown to find all global minima of a combinatorial optimization problem given that the underlying Markov chain is *irreducible*, *aperiodic* and the *detailed balance condition* holds [6].

The standard simulated annealing algorithm basically consists of three steps, 1) the application of a generation function to produce a new solution, 2) the acceptance or rejection of this new solution and 3) the reduction of the temperature.

Using a given probability distribution defined over a set of candidate solutions, the generate function chooses a new solution, j, from a current solution, i. When distances between states are well defined, it may be appropriate to use a generating distribution dependent on the temperature value T_g. When T_g is large, all candidate solutions are equally likely to be generated, thus increasing the probability of generating new solutions far away from the current solution. As T_g approaches 0, the probability of generating new solutions far from the current solution becomes small. Such are the approaches used in Cauchy "Fast" Annealing [9] and Very Fast Simulated Reannealing [3]. The inverse generating function can be used to actually generate new solutions using this distribution [5].

The probability of accepting deteriorations is given by the Metropolis acceptance criterion [2]:

$$A(i,j,T_a) = \begin{cases} 1 & \text{if } E(j) \le E(i) \\ exp(\frac{E(i)-E(j)}{T_a}) & \text{if } E(j) > E(i) \end{cases} \quad i \neq j. \tag{1}$$

Here $A(i,j,T_a)$ is the probability of transition from solution i to solution j given control parameter T_a. As T_a approaches infinity, $A(i, j, T_a)$ approaches 1. Hence all transitions are accepted. As T_a approaches 0, updates that increase the energy function are unlikely to be accepted.

The annealing function describes the rate at which the temperature is decreased. For discrete state spaces, Geman and Geman [13] have proven the statistical guarantee of convergence to a global energy minimum of SA provided the annealing schedule is defined as

$$T_a(k+1) = \frac{T_a(0)}{\ln(k)}.$$

This logarithmic temperature reduction schedule is far too slow for practical implementation. Hence the schedule most frequently implemented is a geometrically decreasing annealing schedule, e.g. $T_a(k+1) = \gamma T_a(k)$ where $0 \leq \gamma \leq 1$.

II. VERY FAST SIMULATED RE-ANNEALING

The main disadvantage of the standard simulated annealing algorithm (Boltzmann annealing) is its slow (logarithmic) annealing schedule. For combinatorial search spaces, this annealing schedule (for the acceptance function) is necessary to guarantee the convergence of the algorithm to the set of globally minimal states. By varying the generating function of the simulated annealing algorithm it is possible to arrive at different "generating" annealing schedules that guarantee convergence to a global minimum for real, compact function spaces. For example, Cauchy, or Fast annealing [9], has a generating function that produces the Cauchy distribution that has a faster annealing schedule than the Boltzmann annealing. The Very Fast Simulated Annealing (VFSA) algorithm [3] is another variant of the standard SA algorithm that has an even faster annealing schedule than the Fast annealing algorithm. Very Fast Simulated Re-annealing (VFSR) adds the concept of *reannealing* to VFSA.

The Very Fast Simulated Reannealing algorithm [3, 4] was developed to search for the state with the lowest function value within bounded and constrained search spaces. This property makes VFSR a good candidate for solving combinatorial image enhancement problems because each optimization parameter (pixel intensity) is bounded by the number of allowable pixel intensities (0 to 255 for an 8-bit gray scale image). Also, in the combinatorial image enhancement task, the solution space is characterized by multiple conflicting constraints that require a solution image be close to the input corrupted image while at the same time adhering to the characteristic property.

The VFSR algorithm uses the generating distribution

$$g_T(\vec{y}) = \prod_{i=1}^{N} \frac{1}{2(|y_i| + T_{gi}) \ln(1 + 1/T_{gi})} \qquad (2)$$

to generate new states [11]. In Equation 2, $g_T(y)$ is the probability density function of generating state $\vec{y}$ $(y_1, ..., y_i, ..., y_N)$ from the current state given a temperature state $\vec{T_g}$ $(T_1, T_2, ..., T_i, ..., T_N)$. Here N is the number of optimization parameters, y_i represent the optimization parameters (e.g. pixel values) and T_{gi} represents the current temperature of optimization parameter y_i, $1 \leq i \leq N$.

The associated very fast annealing schedule is given by

$$T_{gi}(k) = T_{gi}(0) \exp\left(-c_i k^{\frac{1}{D}}\right). \qquad (3)$$

In Equation 3, $T_{gi}(k)$ represents the generating temperature of optimization parameter i at iteration k (i.e. the kth generating state) and $T_{gi}(0)$ and c_i are constants. The annealing variable D is set to N if N is sufficiently small (if $N \gg 1$ the annealing rate is too slow in practice). Otherwise D is set to a small value (e.g. the number of pixel neighbors).

The VFSR generating annealing schedule is said to be very fast as the temperature decays exponentially instead of logarithmically or inversely as a function of k. Using the generating distribution of Equation 2 it has been shown that the exponential generating annealing schedule (given in Equation 3) satisfies the convergence property of basic simulated annealing algorithm [6] (i.e. there exists a nonzero probability of generating any state).

The acceptance function is the Metropolis acceptance function and its annealing schedule is

$$T_a(k) = T_a(0) \exp\left(-c p^{\frac{1}{D}}\right), \qquad (4)$$

where $T_a(0)$, c are constants. Also $T_a(k)$ is the acceptance temperature and p refers to the total number of states that have been accepted during the kth generated state.

A. Reannealing

An integral part of the VFSR algorithm is its reannealing component. The concept of reannealing was introduced by [3] for multivariate function minimization [7]. The idea behind reannealing is to periodically rescale all the SA temperatures. The generating temperature is rescaled with respect to the absolute value of the derivative of each parameter with respect to the cost function. For image processing applications this computation time associated with rescaling each generating temperature is very expensive due to the very large number of optimization parameters. Therefore this aspect of the reannealing algorithm is not considered here.

The acceptance temperature is reannealed by first rescaling the number of acceptances p, based on the logarithmic ratio of the initial acceptance temperature and the current acceptance temperature.

$$p = p_{new} = \ln\left(\frac{T_a(0)/T_a(k)}{c}\right)^D. \quad (5)$$

and then rescaling the acceptance temperature accordingly:

$$T_a(k) = T_a(0)\exp\left(-cp_{new}^{\frac{1}{D}}\right). \quad (6)$$

III. PIECEWISE CONSTANT REGRESSION

An important characteristic in image enhancement is the degree of piecewise constancy in an image. Piecewise Constant (PICO) regression [1] has been successfully utilized to perform intra-region smoothing, preserving important feature while removing corruptive noise.

Let $I_{x,y}$ be a two dimensional array of pixel intensities representing the input corrupted image I where $x = 1, 2, ..., \#rows$ and $y = 1, 2, ..., \#cols$. If a line of constant intensity extends in some direction through a pixel at location (x, y) for at least m pixels, then the pixel (x, y) is said to be PICO-m in that direction. The line may be oriented in four orientations each separated by an angle of 45^o, horizontal, vertical and both diagonals, as shown in Figure 1. A pixel

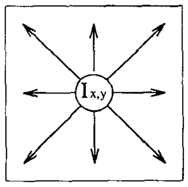

Figure 1: Possible orientation for a PICO-m pixel.

(x, y) belongs to a 4-orientation PICO-m regression if it is PICO-m in all four orientations. Similarly a pixel belongs to a 2-orientation PICO-m regression if it is PICO-m in the horizontal and vertical directions. Since all pixels are PICO-1, the lowest order PICO regression of interest is PICO-2.

To realize a PICO regression, the following energy function should be minimized [1]

$$E(S) = ||S - I|| + C||S - PICO_m||. \quad (7)$$

Here S is the current solution image, I is the original corrupted input image, and C is a constant. $||S - I||$ gives the distance between the input image and the solution image, $PICO_m$ is the closest PICO-m image to S, and $||S - PICO_m||$ is the penalty for pixels in S that are not PICO-m. This energy function basically penalizes the solution image S for differing from the input image I and not adhering to the PICO characteristic property.

Since the PICO regression problem is combinatorial and it uses multi-state optimization parameters to represent a full range of gray levels, the VFSR algorithm should be an efficient method to find a solution image.

IV. APPLYING VFSR TO THE PICO REGRESSION PROBLEM

Figure 2 shows the VFSR algorithm as it is applied to solve the PICO regression problem. Initially the VFSR algorithm sets solution S to the corrupted input image I and the acceptance and generating temperatures are set to a user defined value. The total cost of the initial solution image is calculated using the PICO energy function. Two variables, p and k, that keep track of the total number of acceptances and the iteration number, are initialized to zero.

In the inner loop of the algorithm, each pixel is sampled (one at a time) without replacement (e.g. by random or raster scan). Next the VFSR generating function creates new candidate solutions by perturbing the intensity of each sampled pixel from its previous intensity by generating a random offset using the VFSR generating distribution. The function $Replace(S, x, y, S'_{x,y})$ returns a new image S' by replacing the intensity of pixel (x, y) in solution image S with a new perturbed intensity $S'_{x,y}$. This new solution image S' is a candidate neighboring solution image to S as it differs by exactly one pixel intensity. Based on the acceptance function, S', and hence the new pixel intensity, is accepted or rejected. If accepted, the acceptances counter, p, is incremented by one. Thus $(x_r \times y_c)$ solution images are generated during each iteration for each acceptance and generating temperature values.

The repeated sampling of solution images at constant values of the acceptance and generating temperatures is done to satisfy the SA algorithm's thermal equilibrium condition that the state space be sufficiently sampled at each acceptance temperature [8]. The annealing schedules of the acceptance and generating temperatures are similar. The only difference between the two is that the generating temperature is reduced according to the iteration number k, while the acceptance temperature is reduced according to the number of acceptances p. Because the PICO penalty function depends upon a single pixel and its eight immediate neighbors, the dimensionality, D, in the annealing schedules is set to 9. After a fixed number of iterations in the inner loop, the number of acceptances is rescaled as described in Equation 5. The acceptance temperature is reannealed in the outer loop (6.(c)ii.) based on the rescaled number of acceptances

```
1. Set the initial solution $S = I$
2. Choose initial generating temperature $T_g(0)$ and acceptance temperature $T_a(0)$.
3. Calculate the total energy cost of the image
       $cst = \sum_{x,y} \mathrm{Cost}(S, x, y)$
4. Initialize $k$ and $p$ to zero.
5. Set $R$ to the number of acceptances at which reannealing of acceptance temperature occurs.
6. While the terminating criteria is not satisfied do the following:
    (a) For every pixel $(x, y)$ in $S$ sampled in some order do
        i. Transform current solution $S_{x,y}$ into another feasible solution $S'_{x,y}$ using the VFSR generating function
           $S'_{x,y} = \mathrm{Generate}(S_{x,y}, T_g(k))$
        ii. $S' = \mathrm{Replace}(S, x, y, S'_{x,y})$
        iii. Calculate new cost of the image $S'$
               $newcst = \sum_{x,y} \mathrm{Cost}(S', x, y)$
        iv. Accept or reject the new solution $S'$ based on the acceptance function
            if $(\mathrm{Accept}(cst, newcst, T_a(k)))$ then
                A. $S = S'$
                B. $cst = newcst$
                C. $p = p + 1$
        v. Check for Reannealing : If $p = R$ then
                A. $p = \mathrm{Reanneal}(T_a(0), T_a(k))$
    (b) $k = k + 1$
    (c) Reduce $T_g$ and $T_a$ using the VFSR temperature reduction schedule
        i. $T_g(k) = \mathrm{Anneal}(k, T_g(0))$
        ii. $T_a(k) = \mathrm{Anneal}(p, T_a(0))$
7. Return $S$
```

Figure 2: The VFSR algorithm

in 6.(a)v.A. The VFSR algorithm terminates 1) If the generating or acceptance temperatures falls below a predefined value close to zero, or 2) If the cost in successive iterations repeats more than a predefined number of times.

It is possible to significantly increase the speed of the algorithm. Part of the cost in Equation 7 is determined by summing each pixel's PICO penalty. In step 6.(a)iii., the new cost is calculated using all of the pixels PICO penalties. Because changing pixel (x, y) affects only the PICO penalty of that pixel and its four immediate neighbors (for a 2-orientation PICO-2 regression), only five new penalties need be evaluated.

A. Experimental Tests

A 64x64 'eye' test image from the well known Lena image was used to test the efficacy of the Cauchy and VFSR simulated annealing algorithms, and a greedy algorithm. The greedy algorithm replaces pixel intensities with intensities that will most reduce the PICO cost. The original and noise corrupted images are shown in Figure 3. The original image has large regions of constant intensities and small regions of step edges thus making it a good image to test the algorithms. A Gaussian distributed additive noise model was used to create corrupted images.

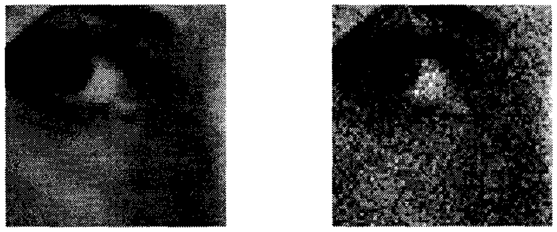

Figure 3: The Original and Corrupted ($\sigma = 20$) 64x64 'eye' images.

All three algorithms were implemented by using random and scan line ordering to modify pixel values. Scan line order generated new pixel values sequentially in scan line order in the image, random ordering generated new pixels randomly without replacement so that each new pixel value was generated once per iteration using Lin's 2-opt method [14]. The sequential scan order worked best for the Cauchy algorithm, and the random scan order worked best for the greedy and VFSR algorithms. The total number of iterations through the entire image was 600 passes. The starting generating and acceptance temperatures were empirically set to 1500 and 500 respectively. The strength constant C was empirically set to 80.

Table 1 shows the results after applying the three algorithms. A large number of runs of each algorithm were tested (100) to ensure statistical significance from the results. The VFSR algorithm almost

Method	**Initial Cost**	**Average Final Cost**	**Best Cost**	**Time (sec)**
Random Greedy	8928.14	1597.71	1534.56	47.85
Cauchy	8928.14	1300.57	1283.25	1049.27
VFSR	8928.14	1052.03	1010.76	1210.52

Table 1: Runs of the random greedy, Cauchy and VFSR algorithms

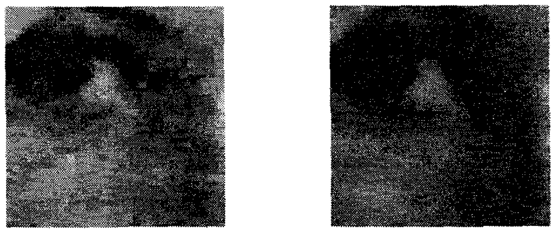
Figure 4: 64x64 Eye image enhanced using a the Random Greedy and Cauchy algorithms.

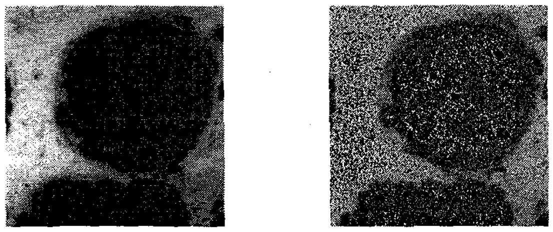
Figure 6: Original and Gaussian noise corrupted ($\sigma = 30$) 256x256 Nerve Cell image.

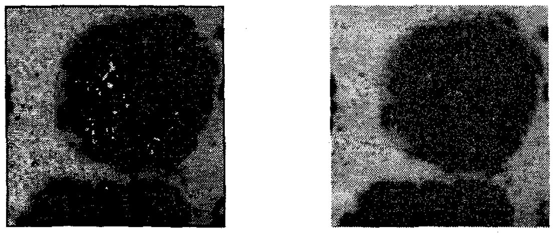
Figure 7: 256x256 Nerve Cell image enhanced using successive 5x5 Median Filters and VFSR algorithm.

always performs better than the other algorithms on this PICO regression image enhancement task.

Figures 4 and 5 show the resultant images obtained after applying the random greedy, Cauchy, and VFSR algorithms to the corrupted eye image. VFSR gave a better PICO filtered image by removing most of the effects of additive noise.

Figure 6 shows the original and corrupted 256x256 nerve cell images. Application of successive 5x5 median filters result in the image shown in Figure 7. The figure also shows the result after applying the VFSR algorithm to the corrupted image. All VFSR parameters were retained from the 64x64 eye image enhancement tests.

Figure 8 shows the cost performance of the greedy, Cauchy, and VFSR algorithms as applied to the eye image. The VFSR algorithm produced the best results for this PICO regression problem indicating that the reannealing component of VFSR has a substantial effect. The greedy algorithm and the Cauchy algorithm converged at a fast rate to a local minimum. VFSR was able to jump in and out of local minima even after a large number of iterations because of the reannealing of the acceptance temperature. Also the quality of the final image obtained after application of the VFSR algorithm was visually more pleasing than the other algorithms as VFSR was able to eliminate 'patch' effects inherent in the greedy algorithm and at the same time able to reduce the noise substantially better than the Cauchy algorithm.

Figure 5: 64x64 Eye image enhanced using the VFSR algorithm.

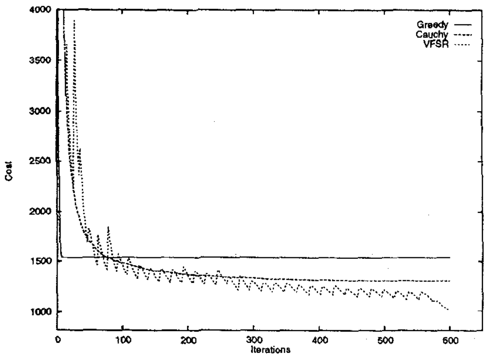

Figure 8: Comparison of cost minimization for the four algorithms.

V. CONCLUSIONS

This paper has shown how advanced simulated annealing techniques can be applied for non-convex combinatorial image enhancement tasks. The effectiveness of VFSR for exploiting local characteristics has been shown using the PICO regression to energy function achieve noise reduction in corrupted images.

Although advanced simulated annealing can be a powerful tool for image enhancement, its suitability for real-time image enhancement problems is restricted due to its high computational cost. Our future work in this area will investigate parallel processor implementation to decrease image enhancement processing time.

VI. REFERENCES

[1] Scott T. Acton and Alan C. Bovik. Nonlinear regression for Image enhancement via Generalized Deterministic Annealing. *SPIE Symposium on Visual Communications and Image Processing*, 1993.

[2] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines *Journal of Chemical Physics*, Vol. 21, No. 6, pp. 1087-1092, 1953.

[3] L. Ingber and B. E. Rosen. Very Fast Simulated Reannealing (VFSR). Technical Report [ftp ringer.cs.utsa.edu: /pub/user/rosen/vfsr.Z], The University of Texas at San Antonio, 1993.

[4] Lester Ingber and Bruce Rosen. Genetic Algorithms and Very Fast Simulated Reannealing: A Comparison. *Mathl. Comput. Modelling*, Vol. 16, No. 11, 1992.

[5] Bruce E. Rosen and Ryo Nakano. Simulated Annealing - Basic and Recent Topics on Simulated Annealing. *Japanese Society of Artificial Intelligence*, Vol. 9, No. 3, pp. 365-372, May 1994.

[6] E.H.L. Aarts and J.H.M. Korst. *Simulated Annealing and Boltzmann machines.* Wiley, Chichester, 1989.

[7] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the "Simulated Annealing" algorithm. *ACM Transactions on Mathematical Software*, 13:272, 1977.

[8] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley. Boltzmann machines: Constraint satisfaction networks that learn. Technical Report Tech. Rept. CMU-CS-84-119, Carnegie-Mellon Univ., 1984.

[9] H. Szu and R. Hartley. Fast simulated annealing. *Phys. Lett. A*, 122(3-4):157–162, 1987.

[10] C. Kittel. Thermal Physics. *John Wiley and Sons*, New York, New York, 1969.

[11] L. Ingber. Simulated annealing: Practice vs. theory. *Statistics and Computing*, (to appear 1993).

[12] S. Kirkpatrick, C.D. Gelatt Jr., and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[13] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution and the Bayesian restoration in images. *IEEE Trans. Patt. Anan. Int.*, 6 (6), 721-741, 1984.

[14] S. Lin. Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal*, 2245-2269, 1965.