COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 45, 269–294 (1989)

# Stochastic Image Pyramids

PETER MEER*

*Center for Automation Research, University of Maryland, College Park, Maryland 20742*

Received August 17, 1987; revised August 8, 1988

A new class of image pyramids is introduced in which a global sampling structure close to that of the twofold reduced resolution next level is generated exclusively by local processes. The probabilistic algorithm exploits local ordering relations among independent identically distributed random variables. The algorithm is superior to any coin tossing based procedure and converges to an optimal sampling structure in only three steps. It can be applied to either 1- or 2-dimensional lattices. Generation of stochastic pyramids has broad applicability. We discuss in detail curve processing in 2-dimensional image pyramids and labeling the mesh in massively parallel computers. We also mention investigation of the robustness of multiresolution algorithms and a fast parallel synthesis method for nonhomogeneous anisotropic random patterns. 

## 1. INTRODUCTION

Image pyramids are versatile data structures employed in multiresolution image analysis. An image pyramid is a hierarchical stack of arrays reproducing the input at decreasing resolution. Each array represents a *level* of the image pyramid and is composed of *cells*. The arrays are mesh connected, a cell having eight neighbors.

Two basic operations are required to generate an image pyramid: low-pass filtering and decimation. The low-pass filtering is necessary to prepare the input image for resolution reduction. It is achieved by convolving the $N \times N$ input with a $K \times K$ kernel having low-pass filtering characteristics. The resulting image is a blurred version of the original with the high resolution details filtered out. The extent of the kernel is much smaller than that of the image.

The filtered image is then resampled, i.e., decimated, to achieve the desired resolution reduction. If the first level of the pyramid should contain $q^2$ fewer cells, only every $q$th pixel is kept along the rows and the columns of the filtered input image. The first level of the image pyramid is thus an array of size $(N/q) \times (N/q)$.

In practice, the amount of computation is significantly reduced by applying the two basic operations in reversed order. First those addresses are selected on the lattice where the filtered image is to be retained for the next level. Then, the weights of the kernel are employed in computing the weighted average in a $K \times K$ neighborhood around each of these addresses. The results are allocated to the corresponding cells at the next level of the image pyramid. A cell at the next level is called a *parent* whenever it is mentioned in relation to the level below. Similarly, the cells in the $K \times K$ neighborhood are called *children* whenever they are mentioned in relation to their parents.

The same sequence of operations can be applied to the $(N/q) \times (N/q)$ array and the second level of the image pyramid is obtained. The new array has only

*The support of the National Science Foundation under Grant DCR-86-03723 is gratefully acknowledged.



0734-189X/89 $3.00

$(N/q^2) \times (N/q^2)$ cells and more of the details present in the input image are lost. Continuing to apply the procedure recursively, after $\log_q N$ steps the apex of the pyramid is reached, i.e., the whole input image is reduced to one cell. In most cases $q = 2$ and $N$ is a power of 2.

Processing in an image pyramid is parallel and recursive. Every parent computes its value independently of other cells on the same level. The parents become the children in the computation of the following level. This leads us to the most important property of the image pyramid: an image pyramid is built in $O$[log(image_diameter)] time. For a review of the subject of image pyramids and their applications see the books of Rosenfeld [23] and Cantoni and Levialdi [3].

Processing in an image pyramid is parallel but not strictly local. Abelson [1] defines a computation as inherently *local* if it can be divided into weakly interacting modules. A computation is inherently *global* if any partition of the process implies substantial interaction among the modules.

Careful analysis of the decimation phase of image pyramid construction shows that this operation does not satisfy the above definition of localness. The next level in an image pyramid is obtained by computing the weighted averages only around the addresses of the form $(j \cdot q, k \cdot q)$ $j, k = 0, 1, 2, \ldots$. This set of addresses defines the *sampling grid* of the next image pyramid level. To perform the decimation process the sampling grid must be generated first. This, however, requires knowledge of absolute positions (coordinates) of the cells in the array. The information is global, all the cells being related to the origin of the array.

At the first sight the globality of the decimation operation does not appear to be a restricting condition. The image pyramids are generated on digital computers and thus any cell implicitly carries its own address within the array. Nevertheless, there are practical problems in which the global information required for decimation depends on the processed feature and is not available. Shift-invariant processing of planar curves in 2-dimensional image pyramids (Section 3.1) and labeling of the mesh in parallel computers (Section 4.1) are discussed as examples later in the paper. We show that these problems cannot be solved in traditional pyramids in logarithmic processing time without having access to the necessary global information during the decimation process. In the following sections we present a new class of image pyramids in which the decimation process makes use only of locally available information. We call the resulting data structures *stochastic image pyramids*. In stochastic pyramids construction of the next level's sampling grid employs random variables.

First, 1-dimensional stochastic pyramids are introduced. The levels of these pyramids are organized in a *doubly linked* list. The doubly linked lists are one-dimensional data structures with a cell being aware (through pointers) only of its two neighbors [9]. Note that this connectivity information is exclusively local. A cell does not know its position within the list. The doubly linked list can be regarded as the 1-dimensional world of an observer walking along a curve. In Section 3.1 we return to this problem in more detail.

A cell in the list carries two pointers indicating connections to its upstream and downstream neighbors along the curve. The cell cannot discriminate, however, which of the neighbors is upstream and which is downstream. This information implies a common global scanning direction and cannot be established locally. As Fig. 1 shows, applying the same local rule leads to a contradiction in global analysis.

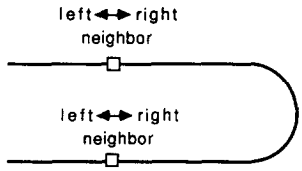



FIG. 1. Inconsistency of the local rules on a global scale. The cells mark their left and right neighbors without knowledge of a common global scanning direction. Backfolding of the curve leads to contradictory results when the local directions are concatenated along the curve.

If both marked cells adopt as scanning direction their locally established "right," global scanning along the curve is ambiguous.

In traditional image pyramids, to generate the sampling grid of the next pyramid level the addresses of the cells within the doubly linked list are necessary. A simple sequential approach takes $O$[list_length] time to establish these addresses. Deterministic methods are available for the recovery in $O$[log(list_length)] processing time of the addresses of cells organized in a linked list. The problem is a particular case of the *parallel prefix* problem [11]. The *distance doubling* algorithm proposed by Wylie [25], in which pointers between more and more distant cells are created recursively, can be employed to solve the parallel prefix problem. The algorithm requires communication between cells through a shared memory and is employed on fine-grained parallel machines like the *Connection Machine* [12]. Cole and Vishkin [4] proposed the *deterministic coin tossing* algorithm to break symmetry situations in parallel environments, a problem equivalent to the recovery of the cells' position in a linked list. The algorithm requires distinct values to be allocated to every cell and achieves the symmetry breaking by manipulation of these values' binary strings.

We are not concerned here with characteristics of the different versions of the above mentioned algorithms, like the speedup achieved and the number of processors required. We note, however, that these deterministic algorithms are contingent upon the existence of a memory shared by all the cells and/or parallel allocation of distinct numbers. Both conditions require implicit involvement of global information. We want to generate image pyramids which can handle inputs in logarithmic processing time even if the cells:

—have no information about their positions in the lattice,

—are connected only to immediately adjacent neighbors,

—cannot distinguish the spatial positions of their neighbors.

Our conditions are much more restrictive than the weakest, exclusive read and write (EREW) model of parallel random access machines (PRAM), employed in the definition of deterministic algorithms. To satisfy our conditions, probabilistic algorithms [20] must be considered. These algorithms employ random variables to settle ambiguous situations but the decisions taken are governed by deterministic rules.

In this paper we approach the problem of image pyramid generation from the viewpoint of probabilistic algorithms. The addresses of the cells within the doubly linked list are not recovered before building a pyramid. While we generate the sampling grids based on local ordering of random variables, configurations very similar to the regular, "traditional" sampling structures are obtained. A new class of

image pyramids, called *stochastic pyramids*, is obtained. It is shown that stochastic pyramids can perform most of the tasks assigned to their traditional (deterministic) counterparts although they satisfy much weaker constraints. Starting from the local information available in stochastic pyramids, recovery of cell addresses and thus reconstruction of deterministic pyramids is possible.

Section 2 treats the 1-dimensional case; Section 3 presents an application to pyramid processing of plane curves and an algorithm for the reconstruction of deterministic pyramids; Section 4 discusses the 2-dimensional case.

## 2. ONE-DIMENSIONAL STOCHASTIC PYRAMIDS

### 2.1. *Assumptions*

Denote the state of a cell $\mathbf{c}$ by a binary number. Let $\mathbf{c} = 1$ when the address of the cell should belong to the sampling grid of the next image pyramid level, and $\mathbf{c} = 0$ when it should not. The binary string of the states thus reproduces the configuration of the sampling grid. In the sequel we assume a twofold resolution reduction between consecutive levels of an image pyramid.

In 1-dimensional deterministic pyramids the sampling grid has the configuration

$$\cdots 10101010 \cdots \tag{1}$$

The state of a cell is determined by inspecting its address within the (1-dimensional) array, becoming 1 if (say) the address is an even number. In stochastic pyramids these addresses are not available. The only allowed global information is:

—the cells have access to independent *identically* distributed (i.i.d.) random variables;

—for all the cells the *same* local decision process is employed.

This global information introduces much weaker constraints than in the deterministic case.

Denote the random variable allocated to a cell by $\mathbf{x}$ and the outcome of a specific draw by $x$. It must be emphasized that our results do not depend on the probability density function of $\mathbf{x}$, $f(x)$, as long as this density is continuous and the random variables are i.i.d. The positivity of the outcomes, i.e., $f(x) = 0$ for $x < 0$, will be found useful later. The condition is not restrictive; the simplest generator for random numbers uniformly distributed between zero and one satisfies it. In stochastic pyramids the state of a cell is established in a few iterations based on the outcomes of the random variables and/or previous states of the cells in a small neighborhood.

A cell can make use of its neighbors only if its neighborhood is symmetrically centered on it. A nonsymmetrical neighborhood implies that the cell can discriminate between the *global* upstream and downstream directions (see Fig. 1 again). Information about global directions, however, is not allowed in a stochastic pyramid. The ideal sampling grid employed for twofold resolution reduction (1) has a periodic pattern of length two. A neighborhood of three cells suffices for generation of the local structure of this sampling grid. We denote the states of the three cells in the neighborhood after the $k$th iteration by $\mathbf{b}(k)$, $\mathbf{c}(k)$, and $\mathbf{d}(k)$. We have avoided the use of the indices $(-1, 0, 1)$ to emphasize that the middle cell $\mathbf{c}$ cannot

discriminate between the spatial positions of **b** and **d** relative to a global scanning direction.

In the local decision process only the state of the cell **c**, on which the neighborhood is centered, can be modified. There is no need for global synchronization among the local processes. If we assume that no cell becomes faulty, i.e., remains in an undecided state, local synchronization can always be achieved by employing "ready" flags.

### 2.2. *Generation of the Initial Sampling Grid*

A cell **c** can employ two different strategies when setting its initial state based on the outcome $x$ of the random variable **x**. In the *discrete case* the cell sets the state to 1 if $x$ is greater than the median of the probability distribution:

$$P_{\text{dscr}}[\mathbf{c}(0) = 1] = \text{Prob}[\mathbf{x} > x_{0.5}] = \tfrac{1}{2}, \tag{2}$$

where the median $x_{0.5}$ is the unique solution of the equation

$$\int_0^{x_{0.5}} f(x)\, dx = \tfrac{1}{2}. \tag{3}$$

The probability that the initial state of the cell will be zero is

$$P_{\text{dscr}}[\mathbf{c}(0) = 0] = 1 - P_{\text{dscr}}[\mathbf{c}(0) = 1] = \tfrac{1}{2}. \tag{4}$$

The discrete case corresponds to the tossing of a fair coin and setting the state of the cell to one if (say) tails appears. Coin tossing is a basic method in randomized parallel algorithms (e.g., [7, 17]). We will prove, however, that for our goal the method is inferior to the other possible strategy.

In the *continuous case* the cell sets its initial state to 1 if the outcome of the random variable $\mathbf{x}_\mathbf{c}$ is the largest in the neighborhood:

$$P_{\text{cont}}[\mathbf{c}(0) = 1] = \text{Prob}[\mathbf{x}_\mathbf{b} < \mathbf{x}_\mathbf{c} > \mathbf{x}_\mathbf{d}] = \tfrac{1}{3}. \tag{5}$$

The value of this probability is proved in the Appendix.

$$P_{\text{cont}}[\mathbf{c}(0) = 0] = 1 - P_{\text{cont}}[\mathbf{c}(0) = 1] = \tfrac{2}{3}. \tag{6}$$

A *run* is a contiguous set of cells having the same state. Thus, $n$ consecutive cells with state 1 constitute a run of length $n$, and will be denoted by $(1)_n$. In order to approximate closely the ideal sampling grid (1), the initial grid generated through any of the strategies should not have high probability for long runs. The decimation ratio $D$ should also be around 2.

The decimation ratio is defined as the total number of cells divided to the number of cells belonging to the sampling grid. That is, the decimation ratio is the inverse of the probability that a cell sets its state to 1. From the viewpoint of the decimation ratio the discrete case is superior. We have

$$D_{\text{dscr}} = \frac{1}{P_{\text{dscr}}[\mathbf{c}(0) = 1]} = 2, \qquad D_{\text{cont}} = \frac{1}{P_{\text{cont}}[\mathbf{c}(0) = 1]} = 3. \tag{7}$$

TABLE 1
Probabilities of $1(0)_n1$ Configurations in the Initial Sampling Grid

| Length $n$ | Discrete case | Continuous case | |
|---|---|---|---|
| | | Experimental | $Q(n)$ |
| 1 | $1.25 \times 10^{-1}$ | $1.33 \times 10^{-1}$ | — |
| 2 | $6.25 \times 10^{-2}$ | $1.11 \times 10^{-1}$ | $1.07 \times 10^{-1}$ |
| 3 | $3.12 \times 10^{-2}$ | $5.75 \times 10^{-2}$ | $5.74 \times 10^{-2}$ |
| 4 | $1.56 \times 10^{-2}$ | $2.22 \times 10^{-2}$ | $2.31 \times 10^{-2}$ |
| 5 | $7.81 \times 10^{-3}$ | $7.08 \times 10^{-3}$ | $7.44 \times 10^{-3}$ |
| 6 | $3.91 \times 10^{-3}$ | $1.92 \times 10^{-3}$ | $2.00 \times 10^{-3}$ |
| 7 | $1.95 \times 10^{-3}$ | $4.50 \times 10^{-4}$ | $4.59 \times 10^{-4}$ |
| 8 | $9.77 \times 10^{-4}$ | $1.00 \times 10^{-4}$ | $0.92 \times 10^{-4}$ |
| 9 | $4.88 \times 10^{-4}$ | $1.80 \times 10^{-5}$ | $1.65 \times 10^{-5}$ |

Nevertheless, the decimation ratio is a global characteristic of the sampling grid and offers no information about how uniformly the cells with state 1 are distributed. The distribution of these cells can only be described if the statistics of the runs are available. That is, we must compute for different values of $n$ the probability of a run having that length. A run of length $n$ implies that $n + 2$ cells have given states. For example, if the run is of 1's the configuration $0(1)_n0$ has to be generated.

The probabilities are easy to compute in the discrete case. Independence of the random variables yields

$$\begin{aligned} P_{\text{dscr}}[1(0)_n1] &= [P_{\text{dscr}}[\mathbf{c}(0) = 0]]^n \cdot [P_{\text{dscr}}[\mathbf{c}(0) = 1]]^2 \\ &= 2^{-(n+2)} = P_{\text{dscr}}[0(1)_n0], \qquad n = 1, 2, 3, \ldots . \end{aligned} \tag{8}$$

All the configurations of same length have the same probability, and there is a considerable chance of obtaining long runs of 1's. The values of $P_{\text{dscr}}[1(0)_n1]$ for $n = 1, 2, \ldots, 9$ are tabulated in Table 1.

In the continuous case, two adjacent cells cannot both have state 1. If a cell has the outcome of a random variable which is a local maximum, its two neighbors no longer can satisfy (5). We can write

$$P_{\text{cont}}[0(1)_n0] = \begin{cases} \frac{1}{3}, & n = 1 \\ 0, & n = 2, 3, \ldots . \end{cases} \tag{9}$$

The computation of $P_{\text{cont}}[1(0)_n1]$ is tedious in the general case. The method of computation and examples are given in the Appendix. The exact values of the probabilities are of little importance for us. We measured them by running 50 trials with $10^5$ cells each on a VAX 11/785 computer. The random variables were uniformly distributed between 0 and 1. The probabilities obtained for $n = 1, 2, \ldots, 9$ are shown in Table 1. A good approximation of these probabilities is given by the

expression

$$P_{\text{cont}}[1(0)_n1] \approx Q(n) = 0.0826\frac{1.61^n}{n!}, \qquad n = 1, 2, \ldots. \tag{10}$$

The values of $Q(n)$ are given in Table 1. The multiplicative constant 0.0826 was determined by making $Q(1)$ equal to the measured probability. While the relation (10) recalls the density function of a Poisson random variable, it should not be concluded that the length of runs of 0's precisely obeys that probability law.

Inspection of Table 1 reveals that runs of more than four 0's have smaller probability under the continuous strategy than under the discrete one. The difference between the two strategies increases with the length of the run. Recall that in the discrete case runs of 1's have the same probability as runs of 0's. In our simulations, the longest run of any value found in the discrete case had a length of 21. The longest run of 0's found in the continuous case had a length of 11.

Once the initial sampling grid is established, a cell modifies (or not) its state based on the local configuration of the neighborhood. The local decision process is not related to the strategy under which the initial grid was generated. The efficiency of the process is measured by how similar the generated grid becomes to the ideal one after a few iterations. The efficiency of the local process is greatly diminished if the initial runs are longer and/or more random. In the sampling grids generated under the discrete strategy any configuration of the same length has the same probability of occurrence. Longer runs are more probable than in the continuous case.

We conclude that the initial sampling grids obtained under the continuous strategy are better inputs into the iterative correction process. These grids have, however, a theoretical decimation ratio of 3 and not the desired 2. We show in the next section that during the iterations used for correcting the sampling grid, while a close approximation of the ideal grid is obtained the decimation ratio is also decreased. In the sequel we consider only sampling grids generated under the continuous strategy.

### 2.3. *Iterations for Correction*

The local processes in 1-dimensional stochastic pyramids are based on neighborhoods of three cells with the cell at the center making the decision about its next state. The three binary numbers representing the initial states of the cells in the neighborhood can be combined into eight *local configurations*. Only five local configurations, however, 000, 001, 010, 100, and 101, can be obtained under the continuous strategy.

Our goal is to reduce the decimation ratio of the sampling grid. To achieve this, the iterative correction process must increase the number of cells belonging to the sampling grid. Similarity with (1) requires that the probability of two adjacent cells both having state 1 should be kept nil. All the local configurations except 000 must be left unchanged during the correction process.

The most direct method suggests that the local configuration 000 should always be turned into 010. Assume, however, a long run of zeros: $1(0)_n1$, $n > 3$. The probabilities of such configurations are given in the second column of Table 1 for different values of $n$. In Table 2 the same probabilities are shown, this time

TABLE 2
One-Dimensional Pyramids: Probabilities of Different Run Lengths, Average Decimation Ratio vs Number of Iterations

| Length | Initial grid | | First iteration | | Second iteration | |
|---|---|---|---|---|---|---|
| $n$ | $1(0)_n1$ | $0(1)_n0$ | $1(0)_n1$ | $0(1)_n0$ | $1(0)_n1$ | $0(1)_n0$ |
| 1 | $1.42 \times 10^{-1}$ | $3.39 \times 10^{-1}$ | $2.89 \times 10^{-1}$ | $4.32 \times 10^{-1}$ | $2.94 \times 10^{-1}$ | $4.35 \times 10^{-1}$ |
| 2 | $1.12 \times 10^{-1}$ | 0 | $1.35 \times 10^{-1}$ | 0 | $1.36 \times 10^{-1}$ | 0 |
| 3 | $5.55 \times 10^{-2}$ | 0 | $0.27 \times 10^{-2}$ | 0 | $< 10^{-5}$ | 0 |
| 4 | $2.07 \times 10^{-2}$ | 0 | $0.18 \times 10^{-3}$ | 0 | 0 | 0 |
| 5 | $6.44 \times 10^{-3}$ | 0 | $0.12 \times 10^{-4}$ | 0 | 0 | 0 |
| 6 | $1.74 \times 10^{-3}$ | 0 | 0 | 0 | 0 | 0 |
| D | 2.951 | | 2.317 | | 2.301 | |

computed for $10^5$ strings of 64 cells each. The probabilities of runs with $n > 2$ are slightly decreased at the expense of increased probability for the shortest 101 runs. This is the effect of truncating at a relatively small base size. There is a significant chance that runs of more than three 0's would occur in the initial sampling grid. Deterministic transition of 000 into 010 turns the configuration $1(0)_n1$ into $10(1)_{n-2}01$. The probability of having $m$ adjacent cells with state 1 is then equal to the probability of an initial configuration $1(0)_{m+2}1$. A probabilistic transition rule, on the other hand, allows the possibility that a long run of 0's is broken into smaller pieces. We employ such a rule.

Let the variable $\mathbf{q_c}(k)$ be defined based on the local configuration after the $(k-1)$th iteration as follows:

$$\begin{aligned} \mathbf{q_c}(k) &= 1, \quad \text{if } \mathbf{b}(k-1) = \mathbf{c}(k-1) = \mathbf{d}(k-1) = 0, \\ \mathbf{q_c}(k) &= 0, \quad \text{otherwise.} \end{aligned} \tag{11}$$

Thus $\mathbf{q_c}(k)$ is the indicator function that a 000 local configuration is present in the neighborhood of $\mathbf{c}$ after the $(k-1)$th iteration. Similarly, the variables $\mathbf{q_b}(k)$ and $\mathbf{q_d}(k)$ can be defined based on the local configurations in their neighborhoods.

At the beginning of the $k$th iteration every cell draws a random number. The following probabilistic rule is employed for the transition of the local configuration 000 into 010:

$$P[\mathbf{c}(k) = 1 | \mathbf{q_c}(k) = 1] = \text{Prob}[\mathbf{q_b}(k) \cdot \mathbf{x_b} < \mathbf{x_c} > \mathbf{q_d}(k) \cdot \mathbf{x_d}] \geq \tfrac{1}{3}. \tag{12}$$

For completeness the rule preserving the other four local configurations is

$$P[\mathbf{c}(k) = \mathbf{c}(k-1) | \mathbf{q_c}(k) = 0] = 1. \tag{13}$$

Rule (12) adapts itself to the local configuration of the five cell neighborhood centered on $\mathbf{c}$. Through the variables $\mathbf{q_b}(k)$ and $\mathbf{q_d}(k)$ the states of cells at distance 2 from $\mathbf{c}$ are also taken into consideration. If the configuration of the five cells after the $(k-1)$th iteration is 10001 then $\mathbf{q_b}(k) = \mathbf{q_d}(k) = 0$. The outcomes of the

random variables x are always positive, and the configuration 10001 will always be changed into 10101. The algorithm deterministically eliminates runs of three 0's. If only one cell with state 1 is present in the five cell neighborhood, i.e., the configuration is either 10000 or 00001, the probability of transition is $\frac{1}{2}$. For an all zero local configuration, the probability of transition is $\frac{1}{3}$, the same as for the initial grid (Section 2.2).

The rule (12) applied in the corrective iterations contains as a special case the rule (5) employed for the generation of the initial grid. By making all $\mathbf{q} = 1$ at the beginning of the sampling grid generation process, when first applied, rule (12) reproduces (5). Thus, the generation of the sampling grid is achieved by repeated application of the same procedure, (12) and (13).

The statistics of the run lengths for the grid obtained after the first correction are shown in Table 2. As expected, long runs of 0's were destroyed and the largest runs now have a length of five. The number of 0's belonging to runs of length one is doubled. The decimation ratio decreased from 2.951 to 2.317.

The statistics of the sampling grid after the second iteration are shown in Table 2. Almost all the runs now have a length of one or two. The residual probability of runs of 0's with length three is less than $10^{-5}$; only four such runs were detected during an experiment with $10^5$ strings of 64 cells each. These runs would be eliminated by a subsequent iteration and in the sequel we do not take 0 runs of length three into consideration. The resulting sampling grid is a good approximation of the ideal grid. The decimation ratio decreased from 2.317 to 2.301.

Only runs with a length of one or two appear in the generated sampling grid. These runs coincide with roots of the correction process and further iterations would not modify the sampling grid. The iteration process thus converged to a stable sampling grid after only two steps. For comparison, we applied similar rules to the initial sampling grids derived under the discrete strategy. The length of the longest run of 1's or 0's decreased to three after six iterations. We conclude that our continuous method for generating the sampling grids of the stochastic pyramids is better.

Somewhat digressing from our main topic, we note that binary strings similar to (1) can be derived from stationary zero mean random fields by recursive differentiation. The zero crossings of the differentiated field tend in the limit to a ...101010... pattern [10]. The number of iterations required to compute the $k$th finite difference locally is $k$ and thus the asymptotic nature of the result reduces its practical importance.

### 2.4. *Structure of the Pyramid*

In stochastic pyramids, just as in traditional ones, the sampling grid determines the structure of the next level. A cell having state 1 after the correction process belongs to the sampling grid. Every sampling grid cell has a correspondent on the next pyramid level and neighbors on the sampling grid become adjacent at the next level.

The cells in the input are organized in a doubly linked list and are aware only of their two immediate neighbors. This local information suffices to generate the sampling grid. Two neighbors in the sampling grid can be separated by at most two cells having state 0 (Table 2). The sampling grid cells thus are not yet connected through pointers to their neighbors on the next level sampling grid. Without these

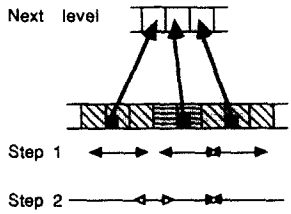



FIG. 2. One-dimensional pyramids: Generation of the next pyramid level. The cells belonging to the sampling grid are marked with black squares. These cells expand their receptive fields in both directions. When the receptive fields of two adjacent cells collide the local structure of the next pyramid level is established by exchanging pointers. The small unfilled circles in the arrowheads show collisions. The receptive fields are highlighted by different hashing.

pointers the next level is not a doubly linked list, and recursive building of the stochastic pyramid is not possible. We now describe a method of defining local connectivity on the next stochastic pyramid level.

The ensemble of cells related to a sampling grid cell will be called the *receptive field* of that cell. Connectivity of the next pyramid level can be established by expansion of the receptive fields. The expansion process takes two steps. A *collision* is detected whenever a cell having state 0 is to be incorporated into two different receptive fields. The sampling grid cell (state 1) tries to incorporate the two adjacent cells (always having state 0) into its receptive field. Detection of a collision stops the expansion of the receptive field in that direction. The two sampling grid cells involved in a collision exchange pointers, and local connectivity on the next stochastic pyramid level is established.

If the distance between two sampling grid cells is one, the collision is detected at the first step when the cell in the middle is accessed simultaneously from both receptive fields. See the collision on the right in Fig. 2. In such a case, the cell with state 0 allocates itself to the sampling grid cell having a larger outcome of the random variable. In the Appendix we argue why such a process cannot lead to a tie. If the sampling grid cells are separated by two cells with state 0 (Fig. 2, left) the collision appears at the second expansion step. Note that while the receptive fields are not necessarily symmetrical, all the operations used are local.

The structure of the stochastic pyramid can now be generated recursively. In Fig. 3 a complete example of a pyramid structure with base length 32 is shown. At each level, the initial sampling grid and the results of the two iterations for correction are given from the bottom to the top. At Level 0, the longest initial run of 0's has length five, but after the corrections the sampling grid employed for the next level has only two runs of 0's of length two. The number of sampling grid cells determines the total number of cells on the next pyramid level. The cells at the ends of the list have only one neighbor. All the rules remain valid for these reduced neighborhoods by making $\mathbf{b} \equiv \mathbf{d}$. The construction of the stochastic pyramid can thus continue up to the apex after only two cells are left (Level 4).

Let the size of the base of the pyramid be $N$. A deterministic pyramid with a twofold resolution reduction between levels has *height*, i.e., number of levels, $\log_2 N$. The almost uniform sampling grid in the stochastic pyramid ensures the logarithmic dependence of the height of the base size. Nevertheless, the probabilistic nature of the processes involved makes the study of the height distribution of interest.

**Level 5**

0

**Level 4**

0 1

**Level 3**

1 0 1
1 0 1
1 0 1

**Level 2**

0 1 0 1 0 1 0
0 1 0 1 0 1 0
0 1 0 0 0 1 0

**Level 1**

1 0 1 0 1 0 1 0 0 1 0 1 0 1 0
1 0 1 0 0 0 1 0 0 1 0 0 0 1 0
1 0 1 0 0 0 1 0 0 1 0 0 0 1 0

**Level 0**

1 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 0 1 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 0 0 0 1 0
1 0 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 1 0

FIG. 3. Example of the structure of a one-dimensional stochastic pyramid with base length 32. At each level the initial sampling grid and the results of the two iterations for correction are shown from the bottom to the top. The number of cells with state 1 determines the total number of cells on the next level.

The probabilities of the height of a stochastic pyramid, given the size of its base, are shown in Table 3. The results are based on $10^3$ trials for each base size. The heights never exceed $\log_2 N$. As the size of the base increases, the expected pyramid height shifts toward $\log_2 N - 1$, in conformity with the higher decimation ratio.

We have computed the average decimation ratio for every level of the pyramid. These values were then averaged again for the lower parts of the pyramid. For the highest levels the averaging introduces artifacts caused by the unequal heights of different pyramids. The results obtained, together with the standard deviation and the number of levels employed, are shown in Table 4. The sampling structure is robust at the lower pyramid levels, as the small standard deviation values show.

## 3. APPLICATIONS OF ONE-DIMENSIONAL STOCHASTIC PYRAMIDS

### 3.1. *Processing of Planar Curves in 2-Dimensional Image Pyramids*

In this section we discuss a first application of the sampling grid generation technique presented above. For more details see Meer, Sher, and Rosenfeld [15].

TABLE 3
One-Dimensional Pyramids: Probabilities of the Different Heights vs Base Size

| Base size | Pyramid height | | | | | |
|---|---|---|---|---|---|---|
| | 3 | 4 | 5 | 6 | 7 | 8 |
| 16 | 0.011 | 0.989 | 0 | 0 | 0 | 0 |
| 32 | 0 | 0.098 | 0.902 | 0 | 0 | 0 |
| 64 | 0 | 0 | 0.314 | 0.686 | 0 | 0 |
| 128 | 0 | 0 | 0 | 0.614 | 0.386 | 0 |
| 256 | 0 | 0 | 0 | 0 | 0.852 | 0.148 |

TABLE 4
One-Dimensional Pyramids: Average Decimation Ratio vs Base Size

| Base size | Levels averaged | D |
|---|---|---|
| 16 | 3 | 2.131 ± 0.099 |
| 32 | 3 | 2.212 ± 0.064 |
| 64 | 4 | 2.228 ± 0.074 |
| 128 | 4 | 2.271 ± 0.047 |
| 256 | 5 | 2.275 ± 0.051 |

The following definition of a curve in a digital image suffices for the purpose of our discussion. A curve is a connected set of cells (pixels) with every cell having at most two neighbors in the set. The above definition restricts the class of curves to those that can be represented by chain codes [6]. Assume a binary input image (with only two gray level values) containing such a curve. We want to build a 2-dimensional image pyramid where each level of the pyramid carries a reduced resolution representation of the input curve.

The problem has practical applications. Let the curve be the bounding contour of a planar shape. In the popular scale-space methods, a curve is described by 1-dimensional intrinsic functions having the arc length as parameter. At every point of the curve Gaussian kernels of increasing spread are convolved with these functions. The convolution outputs are the smoothed versions of the input. Symbolic descriptions of the shape are then derived by mapping local descriptors (such as inflection points) against the scale of the smoothing process [2, 18, 22]. To facilitate tracking of the extrema in scale-space, the number of samples representing the curve is not reduced with the widening of the smoothing kernel. Parallelism of processing is not an issue in the above cited applications. If emphasis is put, however, on fast parallel processing instead of on the derivation of symbolic representations, decimation of the smoother curve versions is desirable [15]. Image pyramids are perfect candidates to perform the smoothing operations.

In Fig. 4a a generic curve is shown superposed over an intermediate-level array in the image pyramid. The curve is partitioned into curve fragments by the cell

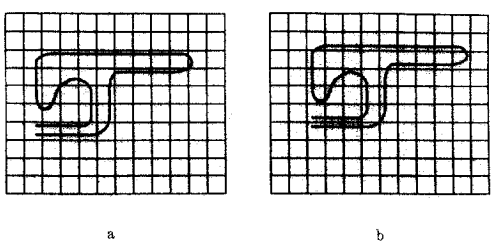



FIG. 4. Planar curve superposed over the sampling grid of an intermediate level in the image pyramid. The curve is only slightly shifted in (b) relative to (a), but the reduced resolution representations would differ drastically.

boundaries. As can be noticed, a cell may have several curve fragments within its boundary and these curve fragments may be located far apart on the curve. Thus 2-dimensional approaches cannot be employed in the resolution reduction process because such methods will fuse the distinct curve fragments together and change the topology of the input.

Fortunately connectivity can always be defined locally across the cell boundaries and this is sufficient to track the curve at any level of the image pyramid. In the input array every cell builds two pointers (indicating connectivity) toward its neighbors along the curve. Note that by employing local connectivity pointers a doubly linked list (Section 1) representation of the curve is built. This representation is distributed across the cells at the base of a 2-dimensional image pyramid. The parents at the first level of the image pyramid employ the pointers of their children to establish connectivity along the curve at the reduced resolution. The new representation is distributed across the first level of the pyramid. The procedure is applied recursively until the apex of the pyramid is reached. The resulting parallel environment, embedded in the architecture of an image pyramid, is called the *chain pyramid* [15].

Beside tracking the curve a parent should also perform the decimation operation. This cannot be achieved in a 2-dimensional image pyramid in a satisfactory way without making use of locally generated 1-dimensional sampling grids for the curve. A process is called *shift-variant* if a shift of the input does not result in the output being shifted by the same amount. Decimation processes are shift-variant [5], and their cumulative effect in an image pyramid is especially severe on curve processing. Compare Figs. 4a and b. The same curve is shown superposed over the array on the same image pyramid level with the curve slightly shifted in Fig. 4b. The small shift causes most of the parents to be allocated different values and the reduced resolution representation of the curve at the next level will change drastically. The recurrent processing in image pyramids only amplifies the effects of the shift at the subsequent levels. The shape of reduced resolution curve representations in the pyramid depend critically on the partition of the curve by the sampling grid. In the worst case even the logarithmic processing time property of the pyramids can be lost [15].

These undesired artifacts of curve processing in 2-dimensional image pyramids can be eliminated if the processing is guided by the doubly linked representations of the curve. As was discussed before, these representations are available on every level of the pyramid. A cell carrying a node of the doubly linked list (that is, belonging to a representation of the curve) can check if any other cell connected to the same parent at next level is part of the list. If not, the cell is called a *siblingless* child. Most of the artifacts of curve processing on image pyramids are caused by the existence of siblingless children.

It can be shown that if a doubly linked list curve representation contains a contiguous string of siblingless children, every other cell in this string can be removed without breaking the connectivity of the next level's representation [15]. The contents of the removed cells are allocated to their neighbors in the list.

If a siblingless child has two "normal" neighbors in the list its removal is easy. However, if long strings of siblingless children are present the selection of cells for removal is equivalent to labeling the string by the sequence $\cdots 0101010 \cdots$. Cells with label 0 are removed. In Section 2 we gave a method of generating such

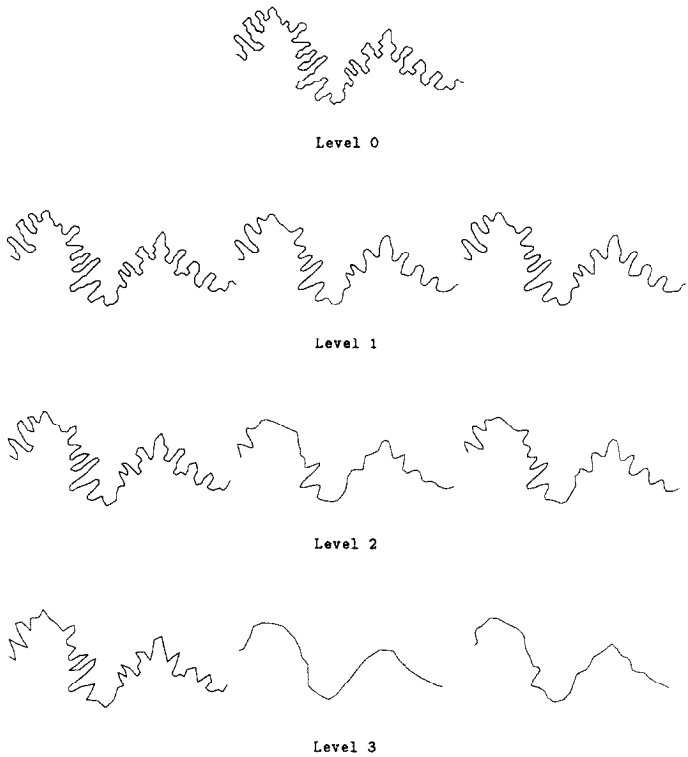



FIG. 5. Curve processing in the chain pyramid. The input curve (level 0) is recursively smoothed by substituting curve fragments with their centroids. Left column: Only the centroids are computed to build the next pyramid level. Middle column: The number of siblingless children is also reduced. Right column: Smoothing is one-dimension, along the curve.

alternating sequences based exclusively on local information. The method does not require the cells' positions in the list and can be employed for labeling of the string of siblingless cells. When two adjacent cells both are labeled with zeros (recall that this cannot happen for three cells) the cell removed is the one with the smaller last outcome of the random variable.

In Fig. 5 the first three levels of an image pyramid are shown performing smoothing of a chain-codable curve. The smoothing is achieved by every parent recursively computing the line centroid for the curve fragment seen within its boundaries. When these centroids are connected by line segments a smoothed curve results. The procedure is described in detail together with other applications of curve processing on image pyramids in Meer, Sher, and Rosenfeld [15]. The representations in the left column are obtained without eliminating the siblingless children. To compute the representations in the middle, the removal procedure for the siblingless children was also applied. Note the drastic improvement in the amount of smoothing achieved. These results are better than the ones obtained when the smoothing was performed in one dimension, *along* the curve, with the arc length as the parameter (the right column of Fig. 5). Note that the latter case is used in scale-space contour processing [2, 18, 22]. The chain pyramid, while through the

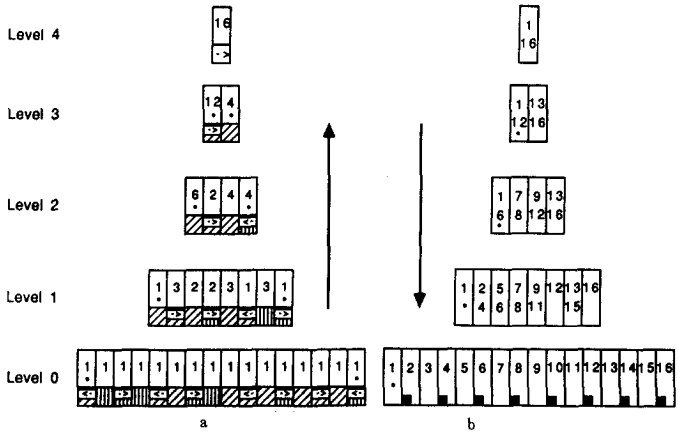



FIG. 6. Reconstruction of the 1-dimensional deterministic pyramid. (a) Bottom-up processes. The sampling grid cells are marked by the insets of the adopted local scanning direction. Cells connected to an end of the input are marked by a dot. The weight of a cell is shown by the number in the box. Adjacent receptive fields have different hashing. (b) Top-down processes. The labels allocated to cells are shown by the numbers in the boxes. The sampling grid at the base of the deterministic pyramid is marked by filled squares.

elimination of siblingless children it simulates smoothing in one dimension, also takes into account the 2-dimensional structure of the curve through the cells performing the processing.

### 3.2. *Reconstruction of the Deterministic Pyramid*

Only local information is allowed during the construction of stochastic pyramids. By exploiting the topology of the levels, however, a 1-dimensional stochastic pyramid can be transformed into a deterministic one. That is, the addresses of the cells within the doubly linked list can be recovered in $O$[log(list_length)] time.

The process of delineating the receptive field connected to a sampling grid cell was discussed in Section 2.4. Assume that every cell in the pyramid carries a *weight*, and let each cell at the input have weight 1. A sampling grid cell on the base of the pyramid can thus count the cells in its receptive field by adding all the weights within the field. The result is the weight of the corresponding cell at the first pyramid level.

In Fig. 6a the receptive fields of a stochastic pyramid (base size 16) are shown by different hashings. The sampling grid cells are marked by the insets with arrows, to be discussed later. Consider the first three sampling grid cells at the left of the base. These cells report to the first level the contents 1, 3, and 2. In the first level's sampling grid they become parts of the same receptive field. When the sampling grid cell on the first level adds up the weights in its field, the weight 6 for a second level cell is obtained. The weight represents the number of cells at the input connected to the cell at the second level.

The process evolves recursively toward the top of the pyramid. The cell at the apex of the pyramid has weight equal to the number of cells in the input, i.e., the size of the base (Fig. 6a). The base size is global information and was obtained from

the structure of the stochastic pyramid. In other applications the cells can perform another associative operation instead of addition. Parallel computation of such global expressions is known in computer science as the *parallel prefix* problem [11]. The partial result of our deterministic pyramid reconstruction algorithm can be regarded as a randomized solution to the parallel prefix problem. We will not diverge here from the main topic of the paper by comparing our method with other randomized solutions or by discussing the number and type of processors required.

The cells on a given level of the stochastic pyramid are organized in a doubly linked list. They cannot discriminate between the spatial positions (upstream, downstream) of their neighbors. Given the 1-dimensional topology of the list the cells know, however, that the two neighbors must be on different sides. This important constraint allows each sampling grid cell to choose a local scanning direction within its receptive field. During the expansion of the receptive field the sampling grid cell correctly connects the cells on either side of it. When the expansion is stopped in both directions, the cell arbitrarily calls one of the expansion directions upstream. The whole receptive field then adopts the established local scanning direction. The insets in Fig. 6a show the local scanning directions adopted by each receptive field in the stochastic pyramid. As expected, these local directions are often contradictory.

The two cells at the ends of the list are aware of their special location because they have only one neighbor. This information is preserved at higher pyramid levels. Cells connected through the structure of the stochastic pyramid to one of the ends of the list are marked in Fig. 6a with a dot.

The operations described until now have employed the stochastic pyramid in a *bottom-up* fashion. The computation starts at the base of the pyramid and continues recursively toward the apex. The size of the pyramid base is computed and the receptive fields are locally ordered. The second part of the deterministic pyramid reconstruction algorithm employs the structure of the stochastic pyramid in a *top-down* fashion. The computation starts from the apex and ends at the base.

The transition from bottom-up to top-down operations is triggered by the cell at the apex establishing a common *global* scanning direction (Fig. 6a, top). This cell also defines the *labels* from 1 to *base_size*, equal to 16 in our example (Fig. 6b, top). The global scanning direction implies that one of the input list ends becomes the first cell of the base. The cell on the level below the apex and connected to this first cell is allocated the labels from 1 to its weight (in our example 12). Recall that the weight carried by a cell represents the number of cells of the input connected to it. The remaining labels are allocated to the other cell (Fig. 6b).

The label allocation process continues recursively toward the base of the stochastic pyramid. The global scanning direction is propagated downward and compared with the local one. When the two directions coincide the labels are allocated in the existing ordering; otherwise, they are allocated in reversed order. A cell divides the received set of labels among its children according to the weights of the children. In our example the weights are 6, 2, and 4 and the scanning directions coincide. The labels are allocated to the first three cells on Level 2 in accordance with the ordering of the receptive field (Fig. 6b). The process stops at the base where every cell now carries a label describing its distance from the starting point.

We have thus succeeded in synthesizing the global information required for the deterministic pyramid. The ideal sampling grid can now be generated by letting

every cell with an even label in the input become a sampling grid cell. Resolution reductions different from 2 can be achieved by using congruence modulo other numbers.

## 4. TWO-DIMENSIONAL STOCHASTIC PYRAMIDS

Assume that at each node of an $N \times N$ square lattice a processor cell is placed. The array is arranged in an 8-connected mesh in which a cell is linked to its neighbors on the north, northeast, east, etc. While the cells can communicate with their neighbors, they cannot localize them in the plane, i.e., they do not know whether a neighbor is on the northeast or on the southwest. The restrictions generalize those in the 1-dimensional case.

Two-dimensional stochastic pyramids can be built starting from strictly local information. As in the 1-dimensional case, the pyramid can then compute global information about the cells. Once the cells are labeled in accordance with their absolute position in the array, simultaneous loading of input information into the pyramid computer is possible. Cell labeling can also be applied to other massively parallel computer architectures [21].

We show that under the most restrictive local conditions labeling of the cells can only be performed in $O[N \cdot \log N]$ time. However, if the conditions are slightly relaxed, allowing the cells to be aware of the relative positions of their adjacent neighbors, the construction of the deterministic pyramid is possible in $O[\log N]$ time.

The key feature in building stochastic pyramids is the generation of the next level's sampling grid. The ideal 2-dimensional sampling grid in a twofold resolution reduction in both coordinate directions (decimation ratio of 4) has the structure

$$\begin{array}{l} 00000000\cdots \\ 10101010\cdots \\ 00000000\cdots \\ 10101010\cdots \\ \cdots\cdots\cdots\cdots \end{array} \tag{14}$$

where the cells with state 1 are preserved for the next level. The ideal sampling grid can be generated by recursively employing the 1-dimensional pyramid reconstruction algorithm described in Section 3.2.

### 4.1. *Jigsaw Reconstruction Algorithm*

All the premises discussed in Section 2.1 for 1-dimensional stochastic pyramids remain valid. The cells have access to i.i.d. random variables with a continuous probability density function. A cell knows the number of its neighbors but cannot localize them on the lattice. The cells on the border are aware of their special positions because they have only five or three neighbors. This fact allows us to reconstruct the grid of (14) under the strictest local information constraints.

Let a connection between two cells be valid only if both cells have *fewer* than eight neighbors. When a cell has two such neighbors, the connection is made to the one with the fewest neighbors of its own. In Fig. 7 an example with $N = 8$ is shown. Application of the connectivity rule results in all the $4(N - 1)$ cells on the border being connected into a 1-dimensional doubly linked list. The 1-dimensional pyramid

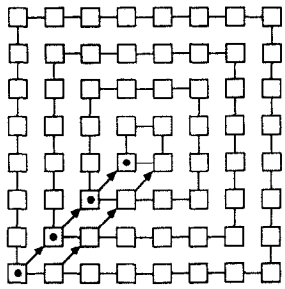


FIG. 7. The jigsaw pyramid. See text for details.

reconstruction algorithm (Section 3.2) can now be applied. The four cells in the corners are the only ones with three neighbors in the array and they mark themselves as candidates for the starting point of the list. Assume that the lower left corner was chosen at the apex of the 1-dimensional pyramid. It is marked with a dot in Fig. 7. The global positions of the cells within the doubly linked list are thus determined in $O[\log 4(N-1)]$ time. The states of these cells can now be defined as in (14).

At the beginning of the next algorithm step the cells on the border mark themselves as already processed. The starting point cell transfers its special role to its only unprocessed neighbor. Then the next cell on the list (with address 2) marks its yet unprocessed and nonstarting point neighbor (Fig. 7). This transfers the adopted global scanning direction to the second *ring* of the array.

Connection from an unprocessed cell to a processed one is now forbidden, and the already processed cells are no longer counted as neighbors. The same connection rule is applied and the second ring of the array containing $4(N-3)$ cells is reduced to a doubly linked list (Fig. 7). The cells in the ring are aware of their distances from the border, by adding one to the distances received from their already processed neighbors. The 1-dimensional pyramid reconstruction takes $O[\log 4(N-3)]$ time to establish the states of the cells as in (14).

The algorithm proceeds recursively toward the center of the array and stops after $[N/2]$ steps with no unprocessed cells left. The total time required to generate the ideal 2-dimensional sampling grid is

$$\sum_{i=1}^{[N/2]} O[\log 4(N-2i+1)] < O[N \cdot \log N]. \tag{15}$$

The algorithm is similar to solving a jigsaw puzzle. First the pieces with at least one straight side are assembled; then the pieces fitting into the already existent frame are sought; etc.

When 2-dimensional problems are approached as a concatenation of 1-dimensional cases, there is extensive use of the natural ordering (the neighbors are always on different sides) in the latter structures. While such algorithms can produce excellent results, the partitioning of the problem into pieces limits the processing time to $O[N \cdot \log N]$. In order to reduce the processing times to $O[\log N]$ the problem must be dealt with in a nondecomposed, 2-dimensional fashion. In Section

4.3 we show that once the natural ordering cannot be employed, reconstruction of the ideal sampling grid is not possible under the strictest local information constraints. Nevertheless, 2-dimensional stochastic pyramids can still perform most of the data manipulation tasks of their deterministic counterparts.

### 4.2. *Two-Dimensional Approach*

The sampling grids of 2-dimensional stochastic pyramids are generated under the same conditions as those of their 1-dimensional counterparts. A cell $\mathbf{c}$ cannot discriminate between the spatial positions of its neighbors $\mathbf{b}_j$, $j = 1, 2, \ldots, 8$ on the square lattice.

Let $\Omega$ denote the 8-connected neighborhood containing the cell $\mathbf{c}$:

$$\Omega: \{\mathbf{b}_0 = \mathbf{c}, \mathbf{b}_j,\ j = 1, 2, \cdots, 8\}. \tag{16}$$

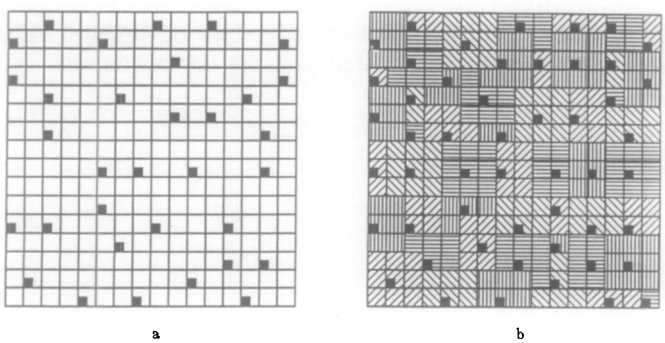
On the border, the definition of $\Omega$ is adjusted according to the number of neighbors. The rules (12) and (13) employed to obtain the 1-dimensional sampling structure can be generalized to two dimensions. To generate the *initial sampling grid*, every cell draws an i.i.d. random variable $\mathbf{x}$. The state of $\mathbf{c}$ is set to 1 if the outcome of the random variable is the largest in $\Omega$:

$$P[\mathbf{c}(0) = 1] = \text{Prob}\left[\mathbf{x}_{\mathbf{c}} = \max_{\Omega} \mathbf{x}\right] = \tfrac{1}{9}. \tag{17}$$

The value of this probability is proved in the Appendix. Note that the maximum based generation rule (17) assures that two cells having state 1 will be separated by at least one cell with state 0. The resulting decimation ratio is

$$D = \frac{1}{P[\mathbf{c}(0) = 1]} = 9, \tag{18}$$

which is much higher than the value 4 obtained for the ideal sampling grid. An example of an initial sampling grid is shown for $N = 16$ in Fig. 8a.



a b

FIG. 8. Example of the 2-dimensional sampling grid generation. (a) Initial grid. (b) The grid after two iterations. The sampling grid cells are marked by the filled squares. The receptive fields in the final grid are delineated by different hashings.

TABLE 5
Two-Dimensional Pyramids: Probabilities of Different Neighborhoods, Average Decimation Ratio vs Number of Iterations

| | Initial | First | Second | Third | Ideal |
|---|---|---|---|---|---|
| $\mathbf{c} = 1$ | | | | | |
| No. of zeros: | | | | | |
| $< 8$ | 0 | 0 | 0 | 0 | 0 |
| 8 | 0.111 | 0.168 | 0.183 | 0.184 | 0.25 |
| $\mathbf{c} = 0$ | | | | | |
| No. of ones: | | | | | |
| 0 | 0.205 | 0.026 | 0.001 | $< 10^{-5}$ | 0 |
| 1 | 0.484 | 0.336 | 0.274 | 0.27 | 0 |
| 2 | 0.185 | 0.379 | 0.418 | 0.42 | 0.5 |
| 3 | 0.016 | 0.085 | 0.114 | 0.116 | 0 |
| 4 | $10^{-4}$ | 0.006 | 0.01 | 0.01 | 0.25 |
| $> 4$ | 0 | 0 | 0 | 0 | 0 |
| $D$ | | | | | |
| Average | 8.994 | 5.943 | 5.464 | 5.439 | 4 |
| S.D. | 0.184 | 0.082 | 0.069 | 0.067 | 0 |

Numerous configurations of $\Omega$ are possible and a precise description of their statistics is unwieldy. We distinguish the neighborhoods by the state of the cell $\mathbf{c}$. The number of neighbors having the *opposite* state are then counted. There are thus two classes of eight neighborhood types each. The probabilities for the different neighborhoods were computed from 100 trials with $N = 64$. The cells on the border were not included.

The measured probabilities for the initial sampling grid are shown in the left column of Table 5. For comparison, the same probabilities for the ideal sampling grid are given at the right of Table 5. Because no two neighbors can both have state 1, all the neighborhoods with $\mathbf{c} = 1$ contain eight 0's and no neighborhood with $\mathbf{c} = 0$ can have more than four 1's. There is a significant probability of empty neighborhoods, i.e., neighborhoods with all the cells having state 0. The average decimation ratio is close to the theoretical 9.

The goal of the *iterations for correction* is to put in the center of each empty neighborhood a cell having state 1 and thus reduce the decimation ratio. Adjacency of two cells with state 1 must be avoided and the correction rule should be also based on local maximum detection.

As in one dimension, we define the indicator function $\mathbf{q}_{\mathbf{c}}(k)$ for an empty neighborhood $\Omega$ after the $(k-1)$th iteration:

$$\mathbf{q}_{\mathbf{c}}(k) = 1, \qquad \text{if } \mathbf{b}_j(k-1) = 0,\ j = 0, 1, \ldots, 8,$$

$$\mathbf{q}_{\mathbf{c}}(k) = 0, \qquad \text{otherwise.} \tag{19}$$

At the beginning of the $k$th iteration every cell draws a random number. To change state, the outcome of the random variable drawn by $\mathbf{c}$ should be the largest among

the outcomes drawn by all those neighbors which also have empty neighborhoods:

$$P[\mathbf{c}(k) = 1 | \mathbf{q}_{\mathbf{c}}(k) = 1] = \text{Prob}\left[\mathbf{x}_{\mathbf{c}} = \max_{\Omega} \mathbf{q}_j(k) \cdot \mathbf{x}_j\right]. \tag{20}$$

The statistics of the sampling grid after each of the first three iterations are given in Table 5. There are no significant changes in the probabilities after the second iteration, and thus as in the 1-dimensional case the correction process converges in only two steps.

Unlike the case of one dimension, however, the resulting sampling grid bears no close similarity to the ideal case. In the initial sampling grid the absence of a natural ordering allows several possible local arrangements among the cells having state 1. These local structures then constrain the generation of further sampling grid cells. A typical sampling grid obtained after two iterations is shown in Fig. 8b. The decimation ratio of the grid is 5.444, which is close to the average 5.464. It must be noted that the average decimation ratio corresponds to two 1-dimensional decimations with ratios 2.38, not far from the value 2.3 obtained in the 1-dimensional case.

The *receptive fields* of the sampling grid cells are delineated by the same expansion process as in one dimension (Section 2.4). The cells expand their fields simultaneously in all eight directions. The expansion in a direction is stopped whenever at least two receptive fields collide. At collision the nonsampling grid cell allocates itself to the sampling grid cell with the largest outcome. All the sampling grid cells involved in a collision exchange connectivity pointers (they are distinguishable by the different outcomes of the random variables) and the structure of the next 2-dimensional stochastic pyramid level is established. The receptive fields in the Fig. 8b example are delimited by different hashings.

Depending on the local order among the outcomes of the random variables the receptive fields have all the possible shapes from one cell to a complete $3 \times 3$ neighborhood. As another effect of the probabilistic sampling grid generation algorithm the 2-dimensional stochastic pyramid levels beyond the base are no longer organized in a regular square lattice. The number of sampling grid neighbors varies and the expected number of neighbors is less than eight. The structure of 2-dimensional stochastic pyramids is analyzed in detail in Meer and Connelly [13]. We conclude that the structure of the stochastic 2-dimensional pyramid becomes increasingly distorted toward random tessellations with fewer connections per cell.

### 4.3. *Applications of Two-Dimensional Stochastic Pyramids*

The receptive fields on every level of the stochastic pyramid cover the array in a compact, nonoverlapping fashion. If the receptive fields are mapped down onto the square lattice of the base (level 0), highly irregular tessellations are obtained. The structure of the 2-dimensional stochastic pyramid can be employed for parallel generation of random patterns [13].

Nonhomogeneous patterns are generated if the tessellation of the base is built with receptive fields taken from several pyramid levels. The hierarchy of the pyramid assures that the different receptive fields combine into a compact covering of the input. Fields chosen from the lower levels yield finely grained random patterns while the ones from the highest levels generate coarse patterns. Anisotropical patterns can be obtained if in (20) weights are also introduced for the pairs

$\mathbf{c}, \mathbf{b}_j$, $j = 1, 2, \ldots, 8$. The preferred orientation of the pattern is then controlled by adequately defining the set of weights. All the computations are performed in parallel and the final pattern is obtained in logarithmic processing time. For examples, see Meer and Connelly [13].

Stochastic pyramids are useful tools in investigating the robustness of pyramidal algorithms under structural perturbations. Two multiresolution algorithms implemented on stochastic pyramids have shown only slight decrease in performance when compared with the results obtained using regular pyramid architecture [14]. In the *bimodality analysis* algorithm, the hierarchy of the pyramid is employed for recursive computation of a statistic. In the *object delineation* algorithm in spite of the irregular pyramid structure the top-down delineation process grows correctly the outline of a blob discriminated at some higher level. Thus, under the most restrictive conditions the stochastic pyramids can be applied to numerous theoretical [16] and practical [24] problems.

Reconstruction of the 2-dimensional deterministic pyramid, i.e., labeling the cells in the base's square lattice with their absolute locations, is not possible in logarithmic time if the cells are not aware of the relative positions of their neighbors. Recall that the jigsaw algorithm described in Section 4.1 makes possible an $O[N \cdot \log N]$ reconstruction.

Let the restriction on local information be relaxed, and a cell be allowed to distinguish among the spatial positions of its neighbors (south, southwest, etc.). The cells are still not aware of their absolute addresses in the array. A local scanning direction (say, west–east and north–south) can now be defined for each receptive field. The sampling grid cells localize their neighbors by keeping track of the direction of receptive field expansion before collision. The 2-dimensional deterministic pyramid reconstruction algorithm is identical under the relaxed constraint with the 1-dimensional algorithm described in Section 3.2.

In a possible extension of stochastic pyramids, generation of the sampling grids can be made adaptive to the weights of the cells. More "busy" regions would have less resolution reduction than smoother parts of the input. Such adaptive pyramids have already been proposed in the deterministic case by Peleg, Federbusch, and Hummel [19].

## 5. CONCLUSION

We have introduced a new class of image pyramids in which only local information is employed to construct the reduced resolution representations of the input. These stochastic pyramids can perform most of the operations their deterministic counterparts are employed for. They can also serve as preprocessors for data structures in which the only available information is local. Global information can be synthesized starting from such inputs and the structure of the traditional pyramid can be reconstructed.

## APPENDIX: COMPUTATION OF THE PROBABILITIES OF $1(0)_n1$ CONFIGURATIONS

In this Appendix we give a method of computing the probability that a run of 0's has length $n$ when generated under the continuous strategy.

Let $\mathbf{x}_j$, $j = 0, 1, 2, \ldots,$ be independent identically distributed random variables with continuous probability density function $f(x)$. We also define the binary

random variables $\mathbf{c}_j$, $j = 1, 2, \ldots$ . The density of $\mathbf{c}_j$ is determined as

$$P[\mathbf{c}_j = 1] = \text{Prob}[\mathbf{x}_{j-1} < \mathbf{x}_j > \mathbf{x}_{j+1}]. \tag{A1}$$

We now compute this probability.

Let the outcomes of the random variables be $x_{j-1}, x_j, x_{j+1}$. The continuous probability density function assures that these three numbers are unequal with probability 1 [8]. (The finite range of numbers which can be represented in a computer introduces only a negligible chance for equality.) We arrange them in ascending order and call the position of the outcome $x_j$ in the sequence its *rank* $r_j$. The value of $r_j$ changes for every new outcome $x_j$, but it is always 1, 2, or 3. We can allocate to $\mathbf{x}_j$ a new discrete random variable $\mathbf{r}_j$ taking only integer values between 1 and 3. The variables $\mathbf{r}_j$ are no longer independent. The sought probability can now be written as

$$P[\mathbf{c}_j = 1] = \text{Prob}\left[\mathbf{r}_j = \max_{k=-1,0,1} \mathbf{r}_{j+k}\right] = \text{Prob}[\mathbf{r}_j = 3]. \tag{A2}$$

The probability of having $\mathbf{r}_j = 3$ is obtained by dividing the number of successes by the total number of possible outcomes:

$$\text{Prob}[\mathbf{r}_j = 3] = \frac{2}{3!} = \frac{1}{3}. \tag{A3}$$

In the general case it is immediate to prove

$$\text{Prob}\left[\mathbf{r}_j = \max_{k=1,\ldots,n} \mathbf{r}_k\right] = \frac{(n-1)!}{n!} = \frac{1}{n}. \tag{A4}$$

We now focus our interest on sequences of $\mathbf{c}_j$ having the form $1(0)_n1$. Based on (A1) it is easy to verify that a sequence $1(0)_n1$ will appear whenever one of the $n$ following *ordering relations* is satisfied:

$$\mathbf{x}_0 < \mathbf{x}_1 > \mathbf{x}_2 \cdots > \mathbf{x}_j < \cdots \mathbf{x}_{n+1} < \mathbf{x}_{n+2} > \mathbf{x}_{n+3}, \qquad j = 2, 3, \ldots, n+1. \tag{A5}$$

As mentioned above, we do not have to be concerned with equalities. The ranks have values between 1 and $n + 4$, and $(n + 4)!$ possible permutations exist. The ranges of ranks satisfying (A5) are overlapping. For example, the ranks $\mathbf{r}_0, \mathbf{r}_{n+3}$ are allowed to take values between 1 and $n + 3$; $\mathbf{r}_1, \mathbf{r}_{n+2}$ between 3 and $n + 4$; and $\mathbf{r}_j$ between 1 and $n + 4 - j$. The computation of the probability of having a sequence $1(0)_n1$ in the general case is beyond the scope of this paper. We will examine only the cases $n = 1$, 2, 3, and 4:

$$n = 1, \qquad \text{Configuration: } 101.$$

Ordering relation:

$$\mathbf{x}_0 < \mathbf{x}_1 > \mathbf{x}_2 < \mathbf{x}_3 > \mathbf{x}_4. \tag{A6}$$

Only the following 16 rank configurations satisfy (A6):

| $r_0$ | $r_1$ | $r_2$ | $r_3$ | $r_4$ |
|---|---|---|---|---|
| 1 | 3 | 2 | 5 | 4 |
| 1 | 4 | 2 | 5 | 3 |
| 1 | 4 | 3 | 5 | 2 |
| 1 | 5 | 2 | 4 | 3 |
| 1 | 5 | 3 | 4 | 2 |
| 2 | 3 | 1 | 5 | 4 |
| 2 | 4 | 1 | 5 | 3 |
| 2 | 4 | 3 | 5 | 1 |
| 2 | 5 | 1 | 4 | 3 |
| 2 | 5 | 3 | 4 | 1 |
| 3 | 4 | 1 | 5 | 2 |
| 3 | 4 | 2 | 5 | 1 |
| 3 | 5 | 1 | 4 | 2 |
| 3 | 5 | 2 | 4 | 1 |
| 4 | 5 | 1 | 3 | 2 |
| 4 | 5 | 2 | 3 | 1 |

This yields

$$P[101] = \frac{16}{5!} = \frac{2}{15} = 0.1333 \tag{A7}$$

which is equal to the experimentally found probability in Table 1.

$n = 2$, Configuration: 1001.

Ordering relations:

$$\begin{aligned} &\mathbf{x}_0 < \mathbf{x}_1 > \mathbf{x}_2 > \mathbf{x}_3 < \mathbf{x}_4 > \mathbf{x}_5 \\ &\mathbf{x}_0 < \mathbf{x}_1 > \mathbf{x}_2 < \mathbf{x}_3 < \mathbf{x}_4 > \mathbf{x}_5. \end{aligned} \tag{A8}$$

When the two ordering relations are combined together we notice that the values of $\mathbf{x}_2$ are not conditioned by $\mathbf{x}_3$ and vice versa. Whatever are the outcomes $\mathbf{x}_2$ and $\mathbf{x}_3$ any of their relative orders satisfy (A8). We can break (A8) into two independent parts. This yields

$$P[1(0)_2 1] = P[\mathbf{x}_0 < \mathbf{x}_1 > \mathbf{x}_2] \cdot P[\mathbf{x}_3 < \mathbf{x}_4 > \mathbf{x}_5] = \tfrac{1}{3} \cdot \tfrac{1}{3} = \tfrac{1}{9} = 0.1111 \tag{A9}$$

which is equal to the experimentally found probability in Table 1.

$n = 3$, Configuration: 10001.

Ordering relations:

$$\begin{aligned} &\mathbf{x}_0 < \mathbf{x}_1 > \mathbf{x}_2 > \mathbf{x}_3 > \mathbf{x}_4 < \mathbf{x}_5 > \mathbf{x}_6 \\ &\mathbf{x}_0 < \mathbf{x}_1 > \mathbf{x}_2 > \mathbf{x}_3 < \mathbf{x}_4 < \mathbf{x}_5 > \mathbf{x}_6 \\ &\mathbf{x}_0 < \mathbf{x}_1 > \mathbf{x}_2 < \mathbf{x}_3 < \mathbf{x}_4 < \mathbf{x}_5 > \mathbf{x}_6. \end{aligned} \tag{A10}$$

Again we can combine the first two ordering relations and have

$$P[1(0)_3 1] = P[x_0 < x_1 > x_2 > x_3] \cdot P[x_4 < x_5 > x_6] + P[x_0 < x_1 > x_2 < x_3 < x_4 < x_5 > x_6]. \tag{A11}$$

This yields

$$P[1(0)_3 1] = \frac{3}{4!} \cdot \frac{1}{3} + \frac{80}{7!} = \frac{29}{504} = 0.05754 \tag{A12}$$

which is equal to the experimentally found probability in Table 1.

$n = 4$. Configuration: 100001.

Ordering relations:

$$\begin{aligned} &x_0 < x_1 > x_2 > x_3 > x_4 > x_5 < x_6 > x_7 \\ &x_0 < x_1 > x_2 > x_3 > x_4 < x_5 < x_6 > x_7 \\ &x_0 < x_1 > x_2 > x_3 < x_4 < x_5 < x_6 > x_7 \\ &x_0 < x_1 > x_2 < x_3 < x_4 < x_5 < x_6 > x_7. \end{aligned} \tag{A13}$$

We combine the first two and the last two ordering relations and notice that all the random variables have the same distribution. This yields

$$P[1(0)_4 1] = 2 \cdot P[x_0 < x_1 > x_2 > x_3 > x_4] \cdot P[x_5 < x_6 > x_7] \tag{A14}$$

which gives

$$P[1(0)_4 1] = 2 \cdot \frac{4}{5!} \cdot \frac{1}{3} = \frac{1}{45} = 0.0222 \tag{A15}$$

which is equal to the experimentally found probability in Table 1.

ACKNOWLEDGMENTS

The author thanks C. Kruskal, S. Peleg, and A. Rosenfeld for suggestions that improved the presentation of this paper.

REFERENCES

1. H. Abelson, Towards a theory of local and global in computation, *Theoret. Comput. Sci.* **6**, 1978, 41–67.
2. H. Asada and M. Brady, The curvature primal sketch, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**, 1986, 2–14.
3. V. Cantoni and S. Levialdi (Eds.), *Pyramidal Systems for Computer Vision*, Springer–Verlag, Berlin, 1986.
4. R. Cole and U. Vishkin, Deterministic coin tossing and accelerating cascades: Micro and macro techniques for designing parallel algorithms, in *Proceedings, Eighteenth Annual ACM Symposium on Theory of Computing, Berkeley, CA, May 28–30, 1986*, pp. 206–219.
5. R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*, Prentice–Hall, Englewood Cliffs, NJ, 1983.
6. H. Freeman, Computer processing of line-drawing images, *Comput. Surveys* **6**, 1974, 57–97.
7. H. Gazit, An optimal randomized parallel algorithm for finding connected components in a graph, in *Proceedings, 27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 1986*, pp. 492–501.

8. J. Hajek, *Nonparametric Statistics*, Holden-Day, San Francisco, 1969.
9. E. Horowitz and S. Sahni, *Fundamentals of Data Structures*, Computer Sci., Rockville, MD, 1976.
10. B. Kedem, *Qualitative Texture Discrimination*, Technical Report CS-TR-1117, Computer Science Center, University of Maryland, College Park, 1981.
11. C. P. Kruskal, L. Rudolph, and M. Snir, The power of parallel prefix, *IEEE Trans. Comput.* **C-34**, 1985, 965–968.
12. J. J. Little, G. Blelloch, and T. Cass, Parallel algorithms for computer vision on the Connection Machine, in *Proceedings, Image Understanding Workshop, Los Angeles, CA, February 23–25, 1987*, pp. 628–638, Kaufmann, Los Altos, CA, 1987.
13. P. Meer and S. Connelly, *A Fast Parallel Method for Synthesis of Random Patterns*, Technical Report CAR-TR-335, Computer Vision Laboratory, University of Maryland, College Park, 1987; *Pattern Recognit.*, in press.
14. P. Meer, S. N. Jiang, E. S. Baugher, and A. Rosenfeld, Robustness of image pyramids under structural perturbations, *Comput. Vision Graphics Image Process.* **44**, 1988, pp. 307–331.
15. P. Meer, C. A. Sher, and A. Rosenfeld, *The Chain-Pyramid. Hierarchical Processing of Contours*, Technical Report CAR-TR-375, Computer Vision Laboratory, University of Maryland, College Park, 1988.
16. R. Miller and Q. S. Stout, Data movement techniques for the pyramid computer, *SIAM J. Comput.* **16**, 1987, 38–60.
17. G. M. Miller and J. H. Reif, Parallel tree contraction and its application, in *Proceedings, 26th Annual Symposium on Foundations of Computer Science, Portland, OR, 1985*, pp. 478–489.
18. F. Mokhtarian and A. Mackworth, Scale-based description and recognition of planar curves and two-dimensional objects, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-8**, 1986, 34–43.
19. S. Peleg, O. Federbusch, and R. A. Hummel, *Custom-Made Pyramids*, Technical Report 255, New York University, Courant Institute of Mathematical Sciences, 1986.
20. M. O. Rabin, Probabilistic algorithms, in *Algorithms and Complexity* (J. F. Traub, Ed.), pp. 21–39, Academic Press, New York, 1976.
21. A. P. Reeves, Parallel computer architectures for image processing, *Comput. Vision Graphics Image Process.* **25**, 1984, 68–88.
22. W. Richards, B. Dawson, and D. Whittington, Encoding contour shape by curvature extrema, *J. Opt. Soc. Amer. A* **3**, 1986, 1483–1491.
23. A. Rosenfeld (Ed.), *Multiresolution Image Processing and Analysis*, Springer–Verlag, Berlin, 1984.
24. A. Rosenfeld, *Pyramid Algorithms for Efficient Vision*, Technical Report CAR-TR-299, Computer Vision Laboratory, University of Maryland, College Park, 1987.
25. J. C. Wylie, *The Complexity of Parallel Computation*, Technical Report TR-79-387, Department of Computer Science, Cornell University, Ithaca, NY, 1979.