#062

CVGIP: GRAPHICAL MODELS AND IMAGE PROCESSING
Vol. 55, No. 5, September, pp. 359–369, 1993

Resolution-First Scanning of Multidimensional Spaces

BLAKE HANNAFORD

Department of Electrical Engineering, FT-10, University of Washington, Seattle, Washington, 98195

Received November 27, 1991; revised April 21, 1993; accepted June 3, 1993

Three methods are introduced for generating complete scans of multidimensional spaces. The traditional method is to use a raster (typically generated by nested iteration) which generates points at the maximum resolution and fills the space slowly. New methods are desirable, because in many applications it is desirable for the scanned points to be distributed throughout the space and for the resolution to increase with the number of points scanned. Three simple methods are introduced in this paper. Two of the methods are members of a class of methods in which the reverse-bit-order operator maps points from "R(esolution)-space" to the desired space. In "R-space" the distance from the origin determines the resolution level of the scanned point. The two scans occupy points in such a way that a distance measure such as the L^1 norm or the L^∞ norm increases with the progress of the scan. The third method uses iteration of primitive polynomials modulo 2 to generate a nonrepeating sequence of binary numbers which eventually fills the space. This method is most computationally efficient, but the L^∞ norm method generates partial scans which completely sample the space at intermediate levels of resolution. Applications are expected in scientific visualization, graphics rendering, multicriterion optimization, and progressive image transmission.

INTRODUCTION

In applications such as multiparameter/multicriteria numerical optimization and multidimensional image processing, a numerical algorithm must be applied to each of the points in a multidimensional array. This algorithm evaluates a function of the point which is a performance index, cost function, or processed image point. For example, in numerical optimization problems with several criteria being optimized over a high-dimensional parameter space, it may be computationally faster to enumerate the whole space to a certain resolution level as opposed to numerically intensive algorithms such as gradient search run several times to completion. In image processing (two- and three-dimensional), a computation based on a kernel must be applied to each point in the image. Typically, these points are scanned in a raster pattern in which points are iteratively visited by indexing each dimension in a nested loop fashion. By this method, the total computation time to cover the multidimensional range of interest is very significant. Furthermore, the sequential iteration of the points means that large features of any underlying form in the data will not emerge until most of the computation is complete. The traditional raster scan has fixed resolution in that the distance between sequential points is constant for most of the points.

It is therefore of interest to investigate other methods for deterministically enumerating all of the points in a multidimensional space which generate a scan whose resolution increases with time. In such a scan, features of the function of interest (or image) would emerge as the scan completes subscans whose resolution is adequate to sample those features. This is reminiscent of the problem of progressive transmission of images (Sloan and Tanimoto [7], Ngan [5], and Tzou [10]) but differs because that work has concentrated on the problem in which it is expensive to transmit (i.e., visualize) a data point, but inexpensive to evaluate it. Thus, the various methods for progressive image transmission such as the discrete cosine transform rely on extensive computation which uses large numbers of points in order to generate each transmitted point. In contrast, in this problem, each point is expensive to evaluate and easy to image. One similarity, however, is that various scans have been used to decide the order of the transmission of transformed points (Ngan [5]). From our point of view, what is required is an algorithm which is (1) computationally efficient, (2) deterministic, and (3) generates all the points in a multidimensional space without repetition and in such a manner that the resolution of the scan increases with time. An additional desired attribute for the scanning algorithm is that it be "state-ful," i.e., that the next scanned point can be generated from the last generated point and perhaps a small amount of state information. Raster scanning, of course, requires only the current point for "state memory."

Deutch [2] studied a variant of the van der Corput sequence in the context of scanning cathode ray tubes with long persistence phosphors.

Another related body of work is the use of quasirandom numbers in numerical integration (Niederreiter [4] and Sobol [9]). In this area it was discovered that certain

1049-9652/93 $5.00

deterministic, patterned sequences ("quasirandom" as opposed to "pseudorandom" numbers) guarantee improved rates of convergence for Monte Carlo integration. The criteria for good quasirandom number generators are similar to those stated above (Sobol [9]).

A significant issue is how to evaluate the "evenness" or "uniformity" of the scanning methods. The numerical analysts use a measure called "discrepancy," D_N (Niederreiter [4]). For the one-dimensional case, let $x_1, x_2, \ldots, x_N$ be a quasirandom sequence of points on the interval [0, 1]. Then

$$D_N = \sup_J \left| \frac{A(J; N)}{N} - |J| \right|,$$

where N is the number of scanned points, $A(J; N)$ is the number of points in the interval J, and J is allowed to vary over *all subintervals* of [0, 1]. The "best" sequence, from the point of view of the rate of convergence of Monte Carlo integration, minimizes this discrepancy from uniformity.

Because of the requirement to enumerate all subintervals of the scanned space, this discrepancy is intractable to compute numerically for even small sized multidimensional problems (Niederreiter [4]).

ALGORITHMS

In one dimension, the problem is easily solved by the method of reverse-bit-order (RBO) counting which generates the van der Corput sequence (Niederreiter [4]). In this method, the operator RBO[n] is defined which evaluates the binary representation of n and reverses the order of the bits. The scanned point is then the point indexed by the binary number RBO[n]. Note that this requires that a maximum value for n be defined in advance (as does a raster scan, at least for the multidimensional case). Ideally, for even scanning, this maximum value must be a power of 2 minus 1. For example, if the maximum is 15, and $0 \leq n \leq 15$, we have the scan enumerated in Table 1 which demonstrates the order in which a 16-sample one-dimensional discretized space may be visualized by a raster and by the van der Corput (RBO) sequence.

The table illustrates the progression of scanned points by both methods. The first point scanned by the reverse-bit-order method ($n = 0$) is at zero of the range and the second ($n = 1$) is in the center of the range. Subsequent points bisect the remaining gaps. At the end of the scan, all points have been reached and no points are repeatedly visited. Of course RBO[n] is quite simple computationally, especially if special hardware is allowed. We can see that the mapping RBO[n] meets the three criteria above.

The difficulty arises in generalizing this process to spaces of multiple dimensions. To generalize the above criteria to the multidimensional case only one detail must be added to criterion (3). To visualize large features of the function, we must increase resolution as closely as possible to uniformly among the multiple dimensions.

One approach might be to define a vector $N = [n_1, n_2, \ldots, n_M]^T$ and address the multidimensional space by rastering the elements of N and then defining RBO[N] = [RBO[n_1], RBO[n_2, ..., RBO[n_M]]^T. This will not meet

TABLE 1
Reverse-Bit-Order Scanning in One Dimension

	"Raster Scan"		"Reverse-Bit-Order Scan"		
n	b[n]	Points Scanned	RBO[n]	b^{-1}[RBO[n]]	Points Scanned
0	0000	*.....	0000	0	*.....
1	0001	x*.....	1000	8	x.....*.....
2	0010	xx*.....	0100	4	x...*.....*.....
3	0011	xxx*.....	1100	12	x...x...*.....*...
4	0100	xxxx*.....	0010	2	x.*.*...x...x...
5	0101	xxxxx*.....	1010	10	x.x.x...x.*.x...
	...				
12	1100	xxxxxxxxxxx*...	0011	3	xxx*xxx.xxx.xxx.
13	1101	xxxxxxxxxxxx*..	1011	11	xxxxxxx.xxx*xxx.
14	1110	xxxxxxxxxxxxx*.	0111	7	xxxxxxx*xxxxxxx.
15	1111	xxxxxxxxxxxxxx*	1111	15	xxxxxxxxxxxxxx*

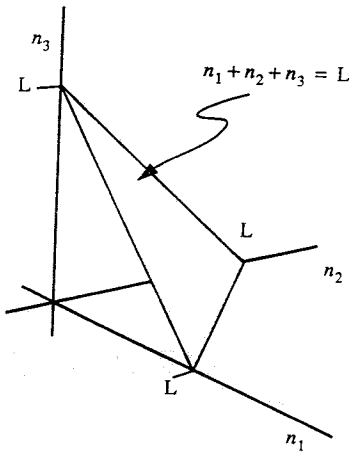

FIG. 1. Three-dimensional example illustration of one of the hyperplanes scanned by the L^1 scan of an abstract space called "R-space." As "L" is increased, hyperplanes are successively further from the origin. Each component of the points in R-space is mapped to the task space by the reverse-bit-order operator (see Table 1).

the modification of criterion (3), because the resolution in RBO[n_1] will increase all the way to the limit before it increases beyond the first point in RBO[n_2], . . . ,RBO[n_M].

To solve this problem we define "R-space" as the space spanned by the vector N. We need a method of scanning R-space which, when mapped to the target space "T-space" by RBO[n] will meet the three criteria. Since the base 2 log of n_i roughly maps to the resolution in dimension i, to meet the modified third criterion, we need a scan of R-space in which some norm $|N|$, increases uniformly with the number of points scanned.

Two methods based on this approach are discussed in the remainder of the paper: hyperplane scans and shell scans. These correspond to using the L^1 norm and the L^∞ norm, respectively. They are compared with the raster scan and with each other in a two-dimensional example. The algorithms (detailed in the Appendices) work for any number of dimensions.

An L^1 scan of R-space starts with the origin and scans hyperplanes in which $n_i = L$, for successive values of L. This method thus reduces to finding all n-tuples with $n_i > 0$ whose sum is L. The L^1 scan is conceptually illustrated for the three-dimensional case in Fig. 1.

As a two-dimensional example, consider the function

$$z(t_1, t_2) = \begin{cases} 1 & r1 < r < r2 \\ 0 & \text{otherwise} \end{cases}$$

$$r = \sqrt{t_1^2 + t_2^2}.$$

This "donut function" can be visualized by scanning two-dimensional T-space through the L^1 method and plotting points where $z(t_1, t_2) = 1$. The L^1 method was used to scan this function with 64 possible grid values of each of t_1 and t_2 for a total of 4096 sample points. The RBO[n] operator was set for 8 bits, giving output values between 0 and 255. For comparison purposes, T-space was also scanned by a conventional 64 × 64 raster. Figure 2 illustrates the built-up visualization after 1%, 6% (n = 256, see below), 10%, and 50% of the 2^{12} points have been evaluated with both the raster scan and the R-space L^1 scan.

By the time 10% (410) of the 4096 grid points have been sampled by the R-space L^1 method, the "donut" pattern is clearly visible. Qualitative features that can be discerned with high probability at this level include that the shape is connected and that it encloses a hollow region.

In contrast, when the raster scan has completed 10% of the grid, no conclusions about these qualitative features can be drawn. Even at the 50% scan level, the raster scan allows no conclusions to be drawn about, for example, whether the shape encloses the hollow region.

A subtlety of the L^1 method is that since the hyperplanes are angled with respect to the R-space axes, the resolution will always be highest along the directions of the axes, and lower in between. This is illustrated by the partial scan (Fig. 2h) which is completed to 50% of the 64 × 64 = 4096 possible grid points. This corresponds to the hyperplane with the sum $L = 63$. Note that certain horizontal and vertical lines are sampled at higher resolution than the spaces in between them.

To eliminate this problem at the end of the scan, we can constrain the hyperplane to a box. At the end of the scan, we want to cover all points N such that $n_j < n_{\max}$ $\forall$ $(j < M)$.

This means that the hyperplanes must be truncated when $L > n_{\max}$. A computer program to generate these L^1 scans is outlined in Appendix A.

When the L^1 scan is constrained as described above, and if $n_{\max} = 2^j - 1$, the end result of the scan is a set of evenly distributed points on the T-space. However, during intermediate parts of the scan, the sloping hyperplanes in R-space generate nonuniform intermediate scans. Thus, resolution increases more in directions close to one of the principle axes and criterion (3) is not fully satisfied. Although this effect is relatively mild in the two-dimensional example shown, it gets worse as the number of dimensions increases.

One way to describe this phenomenon is to see that as M, the number of dimensions, increases, the "cross-terms," i.e., hyperplane points where

$$\max [n_i] \ll L,$$

get progressively closer to the origin.

A possible solution to this problem is provided with the second method of scanning R-space, "shell scanning,"

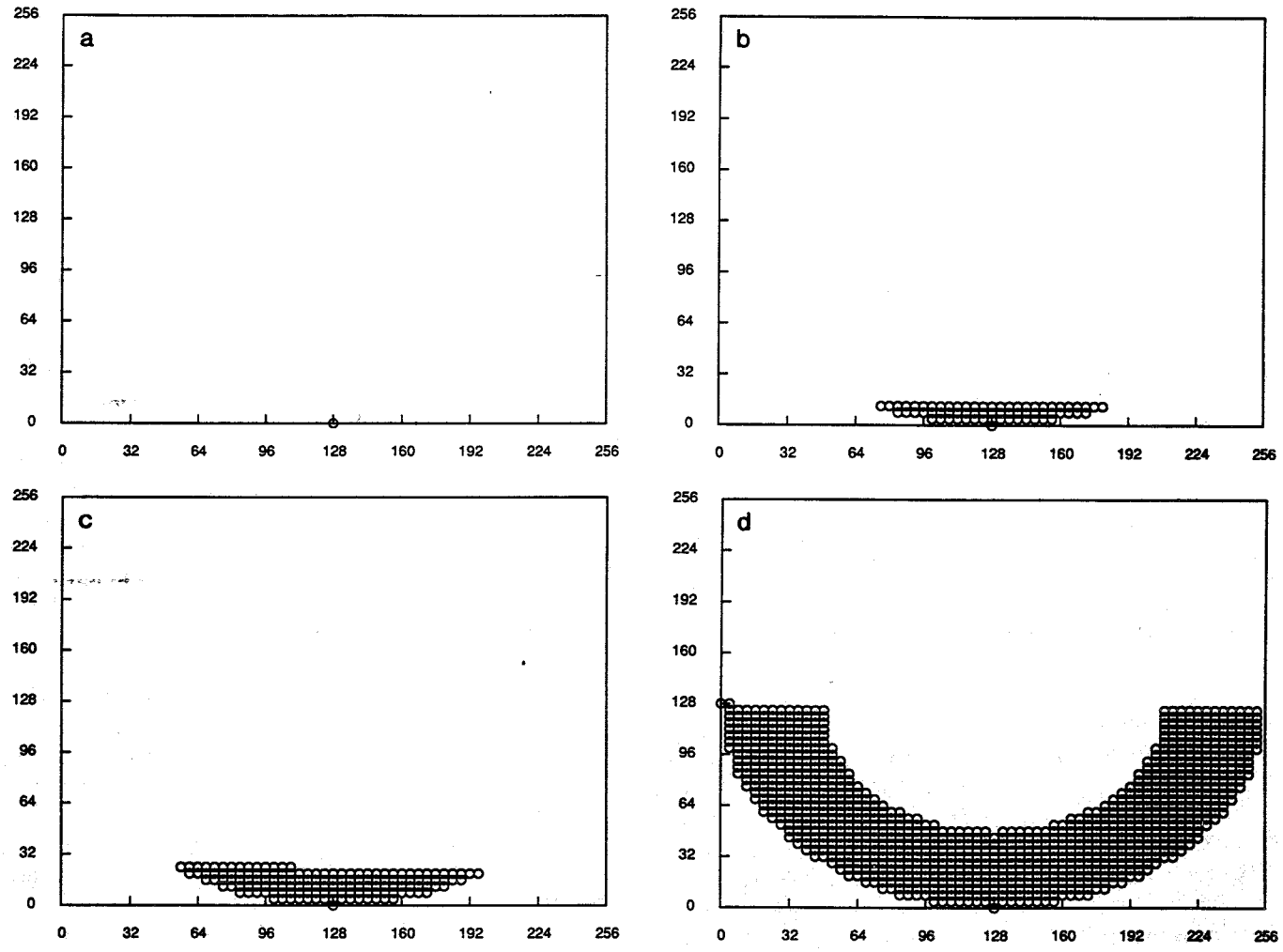

FIG. 2. Two-dimensional visualizations of the "donut" function (see text). The space is discretized into 64 × 64 = 4096 points. Scans are partially completed with (a) 1%, (b) 6%, (c) 10%, and (d) 50% of the total number of points. Although resolution is high, coverage of the space is low, which limits information about "big" features such as the donut from partial scans. Also shown are visualizations using the L^1 scan for the same numbers of points (e–h). Note that the donut shape is quite clearly distinguished after only 6% (f) of the points have been scanned.

in which the L^∞ norm is substituted for the L^1 norm as the variable which is indexed in R-space. In this method, scanning R-space from the origin out is accomplished by scanning a series of "shells" having unit thickness. The set of points belonging to the jth shell is the set of all R-space points, N, which satisfy

$$|N|^\infty = \max [n_i] = L.$$

In three dimensions these shells (Fig. 3) consist of three sides of a cubical box with L points on a side. By scanning R-space in this fashion, resolution increases in all components to level L when each shell is complete and constitutes a complete sampling of the space for $L = 2^j$, ($j = 1, 2, 3, \ldots$). An algorithm for the R-space shell scanning method is presented in Appendix B. The sketch of Fig. 3 suggests that the three-dimensional shell is actually composed of segments of subspaces of dimension 0, 1, and 2 (outlined in dotted lines). The subspaces are parallel to the axes of R-space and are defined by setting certain components equal to a constant. The L^∞ scanning method is a recursive algorithm which divides the shell into progressively smaller subspaces. Subspaces of dimension one are scanned by counting. The dimension-zero subspace is simply returned as a point.

The L^∞ scanning method was used in two dimensions to scan the donut (Fig. 4). With the L^∞ scan, the density of points builds up in a different fashion than with the L^1 scan. In contrast to the L^1 scan, the L^∞ scan leaves more open lines containing no scan points at intermediate numbers of scanned points (compare, for example, Fig. 4c with Fig. 2g). Furthermore, since at the lowest recursion level (the one-dimensional subspace; see Appendix B) the L^∞ method scans along a single dimension, the L^∞ scan

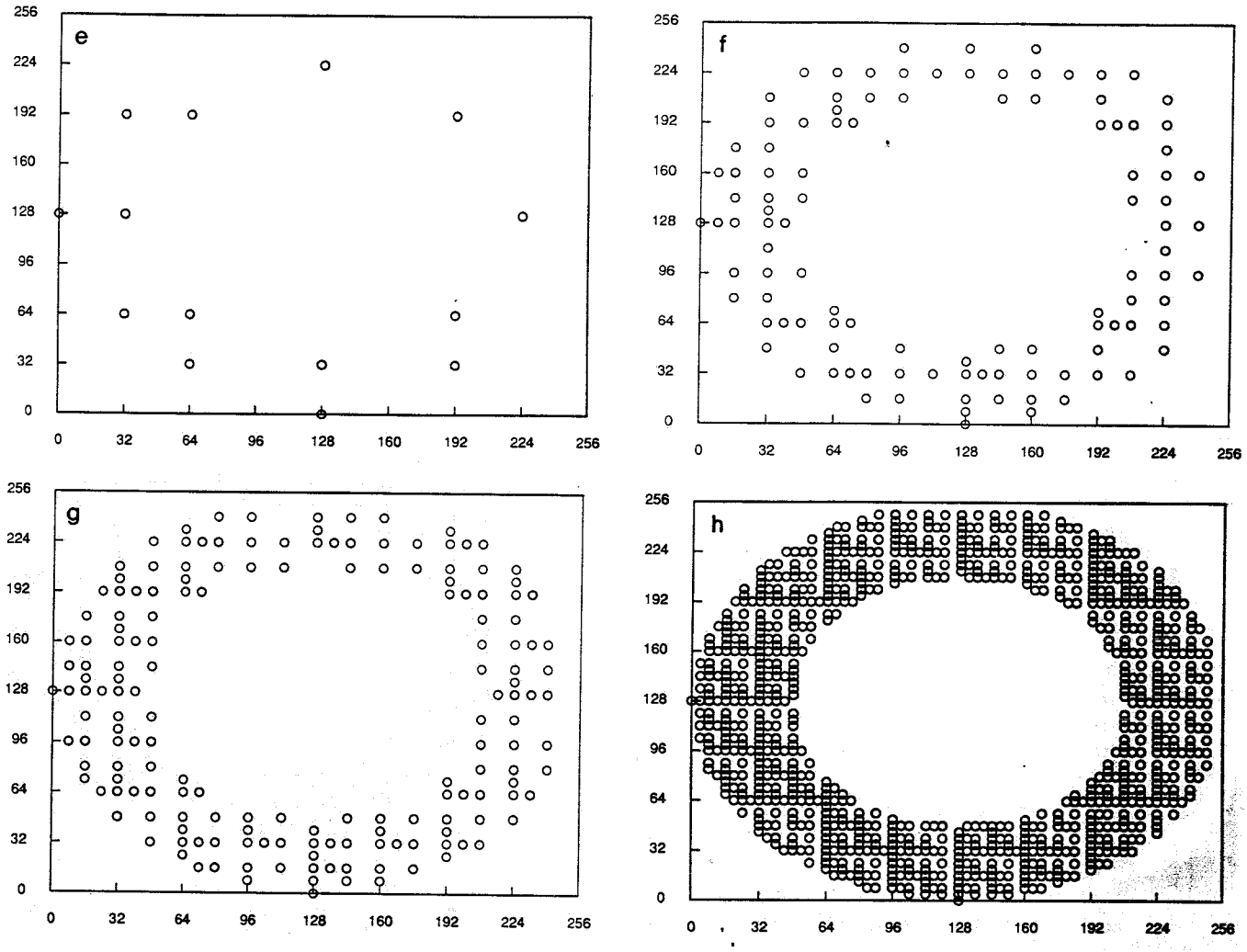

FIG. 2—*Continued*

spends considerable time working on lines parallel to the axes (Fig. 5a). In contrast, the L^1 scan scans along diagonal lines (Fig. 5b). These effects are quite subjectively apparent if the points are plotted sequentially on a graphics screen.

A further property of the L^∞ scan is seen when the number of points is equal to a power of 2 raised to the M power. At these levels (see Fig. 4b, Fig. 6) the L^∞ norm produces an even scan filling the space at a uniform resolution. These uniform sub-scans are referred to as P_τ nets (Sobol [8]). A sequence such as the L^∞ scan which generates all P_τ nets in increasing order is referred to as an LP_τ sequence. In contrast, there are no such levels for the L^1 scan (see Fig. 2f).

Additional Method

A final method to be discussed is the generation of a nonrepeating pseudorandom sequence of binary numbers

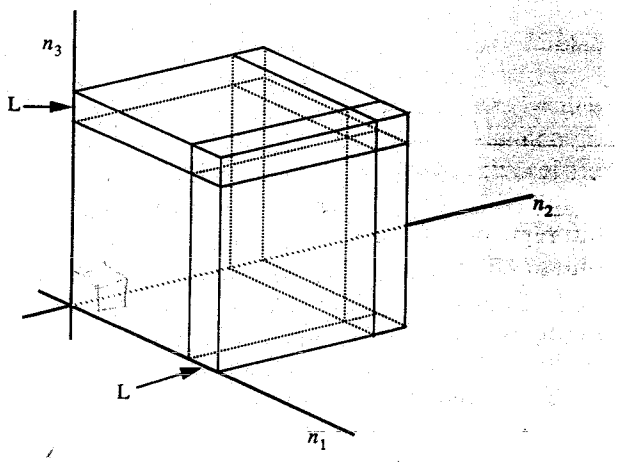

FIG. 3. Another method of scanning R-space, the L^∞ scan, builds up the scan through "shells" of unit thickness (shown here in three dimensions). Each point in the shell is mapped through the RBO operator, then "L" is increased.

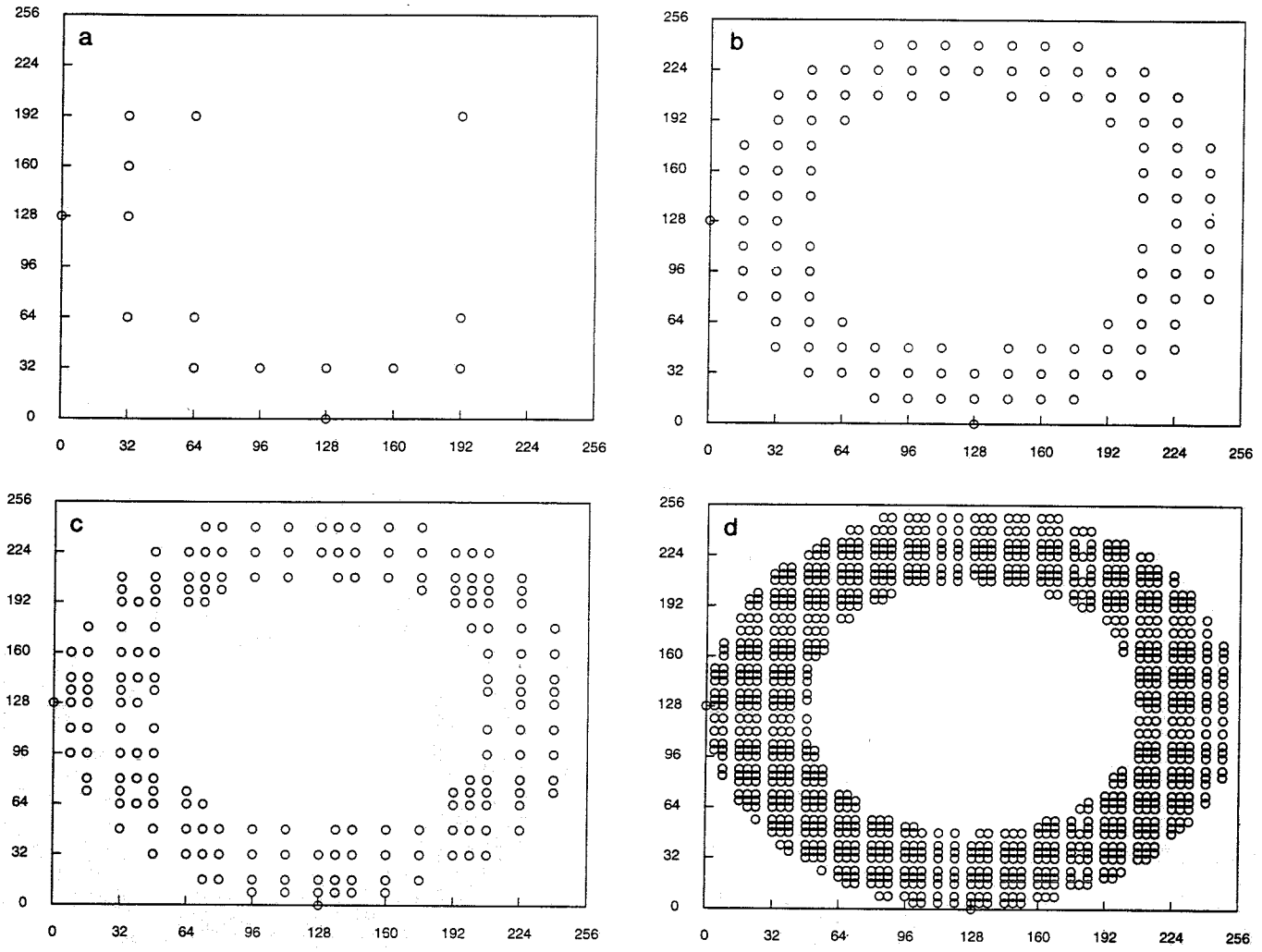

FIG. 4. The L^{∞} method was used to scan the donut function for the same intermediate scans (a–d). Scans are qualitatively different from those of the L^1 method (see text). In particular, for 256 scanned points (b) note that the scan has achieved complete sampling of the space at an intermediate, 16 × 16 level of resolution.

(Baldick [1]) through the iteration of primitive polynomials modulo 2 (PPM2) (Press *et al.* [6]). PPM2's are polynomials whose coefficients are either 1 or 0. Certain of these polynomials of order N_P generate all possible binary numbers less than 2^{N_P}. This sequence is commonly generated by using the exclusive-or operator on selected bits of a shift register to generate a new low-order bit. Successive values are generated by left shifting.

For scanning, the order of the PPM2 can be selected by $N_P = Mj$, where M is the number of dimensions and j is the number of bits to represent the resolved levels in each component. Then each component's scanned value can be taken from a region of the register. For example, in visualizing the donut function, $M = 2, j = 6$, and $N_P = 12$. The scanned values are thus

$$t_1 = b^{-1}[\text{bits}\,[0 - 5]]$$

$$t_2 = b^{-1}[\text{bits}\,[6 - 11]].$$

This method was used to scan the donut function for the same levels of completion as in Figs. 2 and 4 (Fig. 7). The PPM2 scan generates a pseudorandom scan pattern, but leaves gaps and concentrated areas in the partial scans, and does not generate complete rasters of partial resolution. The build up of the PPM2 scan is pleasingly scattered, as seen in the plot of intermediate scanned points (Fig. 5c).

CONCLUSION

We have presented three new, computationally simple algorithms for generating a scan of the points in a discrete multidimensional space such that sampling density increases with the number of points scanned. The three methods, L^1 norm and L^{∞} norm scanning, and scans based on primitive polynomials modulo 2, were contrasted with traditional raster scanning. All three methods produced

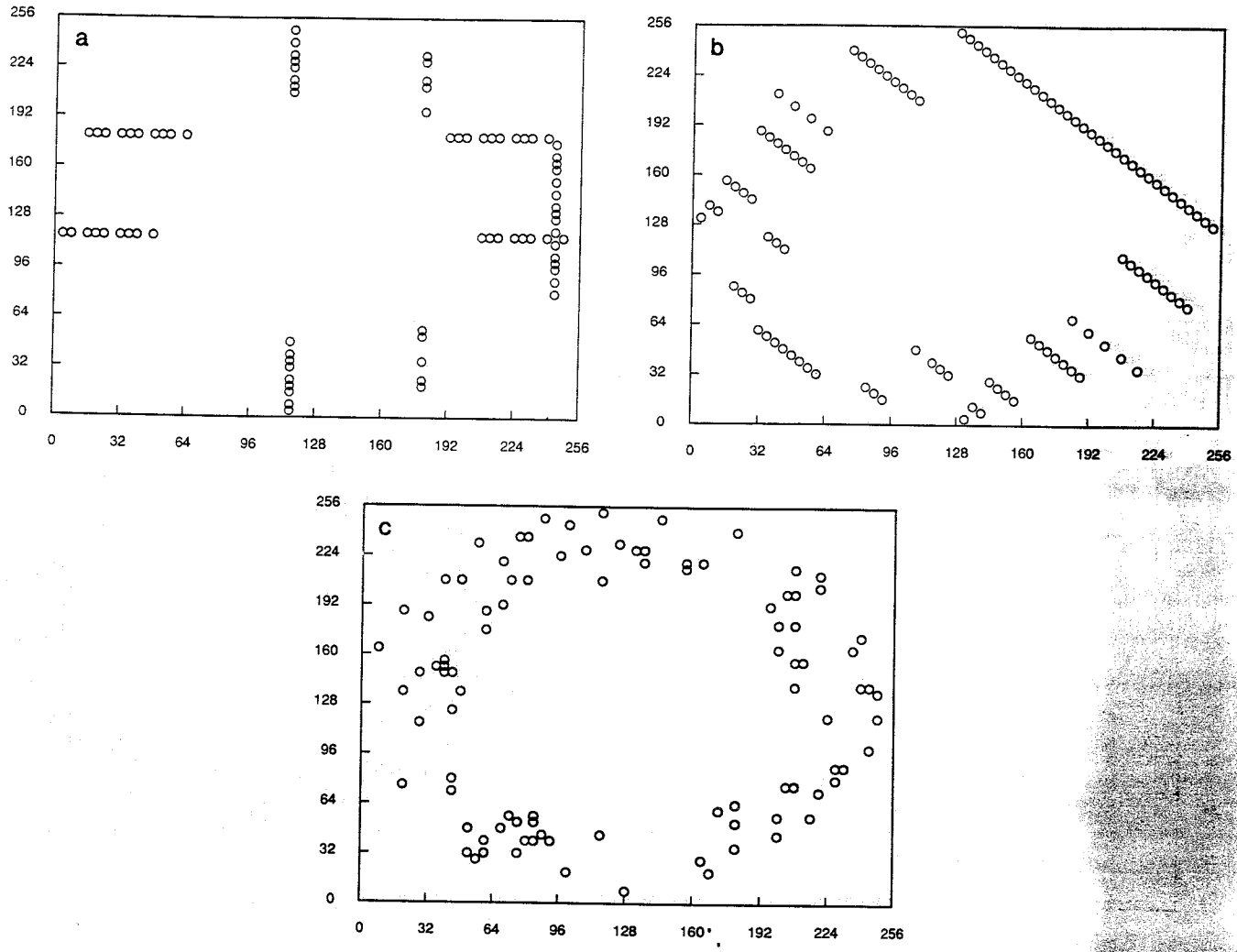

FIG. 5. Partial scans for a range of points from $n = 2048$ (50%) to $n = 2248$ (55%). Note that the L^∞ scan (a) concentrates on selected vertical and horizontal lines while the L^1 scan (b) is concentrated on diagonals. PPM2 scan (c) appears to be scattered in a pseudorandom manner.

increasing resolution with time, and eventually fully sampled the space without repetition. All meet the three criteria proposed above in that they are computationally efficient, deterministic, and generate all of the points in the discretized space without repetition with gradually increasing resolution. The simplest computationally is the PPM2 method, followed by the L^1 norm scan. The efficiency of the L^∞ method is slightly less because it uses recursion. However, the recursion depth is limited to the number of dimensions, and the required state storage per recursion is small. The L^1 and PPM2 methods generate scans which subjectively appear uniform in terms of the "scatter" between sequential points, but generate no intermediate P_r scans as does the L^∞ scan (Fig. 6). However, the L^∞ scan dwells on single lines and thus generates subsequences of points which are locally clustered on lines parallel to one of the axes. The PPM2 scan generates a pseudorandom pattern which has locally uneven regions of scan density, which may be bad for certain tasks.

These three methods have application to multidimensional optimization (Gorbunov-Possadov *et al.* [3]), visualization, multidimensional imaging, and other problems. Similar algorithms may be generated to scan according to intermediate norms (L^2 and up) if efficient algorithms can be found.

With regard to the image processing applications in particular, the L^∞ scan, *without the reverse-bit-order transform,* would have a possible use in progressive transmission of images that are compressed with the discrete cosine transform. Ngan [4] has used a method essentially the same as the L^1 scan to determine the order of transmission of these coefficients. However, this scan emphasizes high-frequency horizontal and vertical lines in the image. Replacing the L^1 scan with the L^∞ scan should result in a smoother increase in detail, as well as intermediate images which contain all information below the current spatial frequency. This should result in better overall image quality earlier in time.

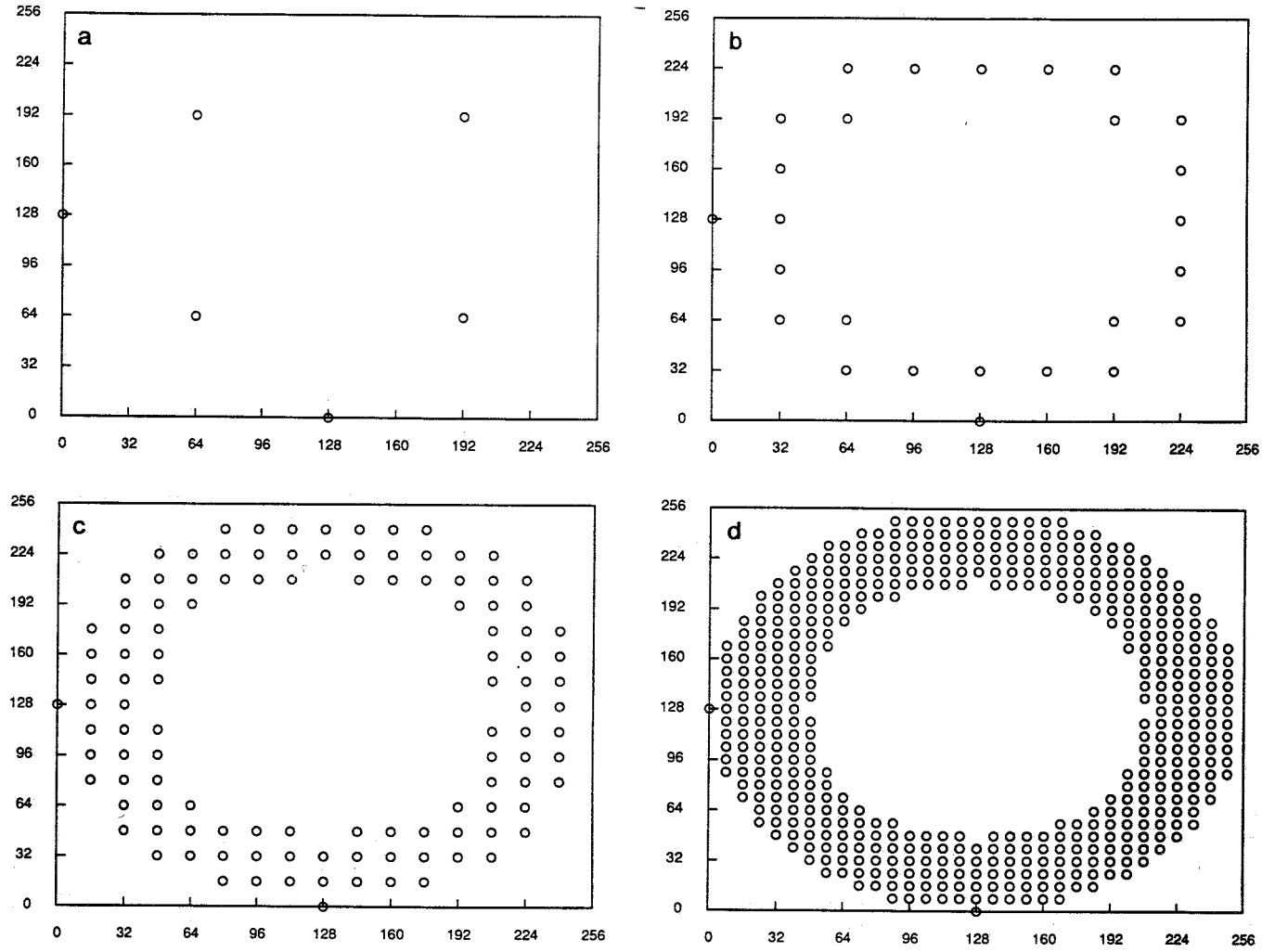

FIG. 6. Complete L^{∞} scans at intermediate resolution levels (a) 4 × 4, (b) 8 × 8, (c) 16 × 16, and (d) 32 × 32.

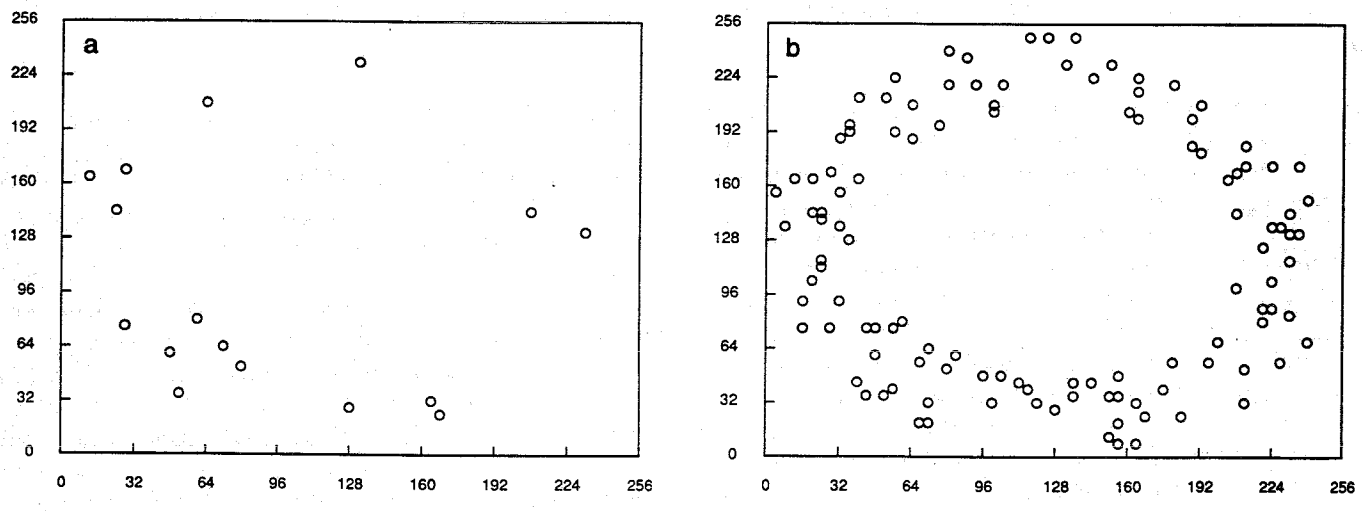

FIG. 7. Two-dimensional scans (a–d) generated by primitive polynomial modulo 2 (PPM2). This scan generates pseudorandom binary numbers in a nonrepeating sequence which are used directly on the task space. This method can be implemented in hardware with only a shift register and a few exclusive-or gates. These scans are illustrated to the same "depth" as those in Figs. 2 and 5.

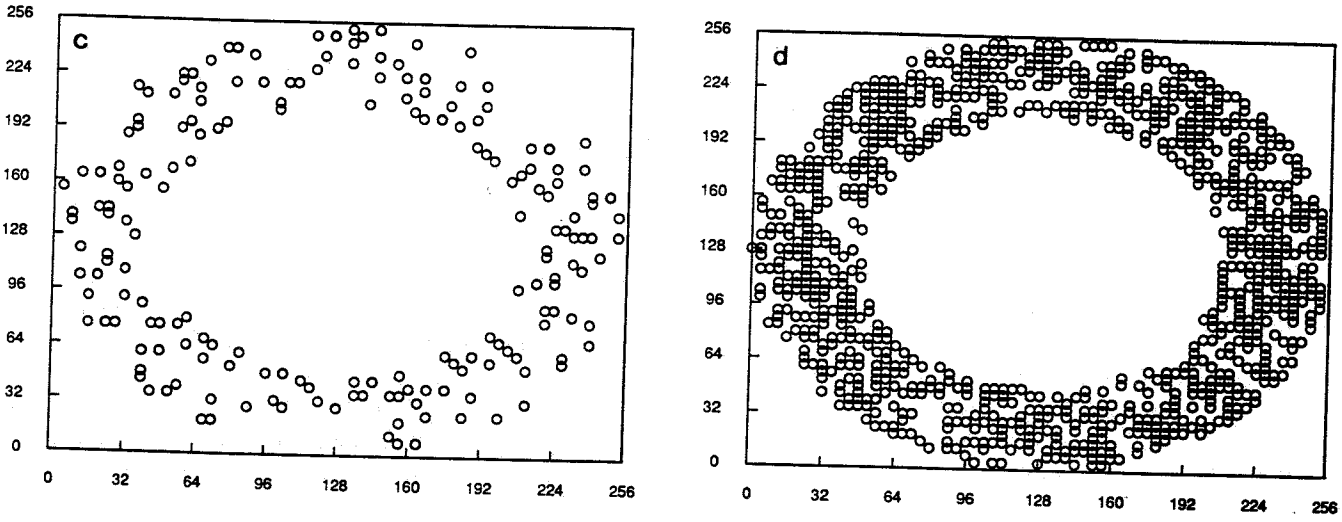

FIG. 7—*Continued*

APPENDIX A: ALGORITHM FOR HYPERPLANE (L^1) SCANNING

This algorithm[1] successively enumerates all of the vectors, `N [ ]` containing elements `N[i] <= ''nmax''` whose sum is `''L.''` The "level pointer," `P`, keeps track of the level in `N[ ]` at which enumeration is taking place. Each time `''next (L)''` is called, it returns a new vector `N[ ]` whose elements sum to `L`. `N[ ]` should be initialized to contain zeros.

```
next (L)
{
tot = bump = 0
for(j = Nd-1 ; j >= 0 ; j = j - 1 ) {
                if (j < P) {
                     N[j] = min(L - tot, nmax)
                     }
                if (j = P) {
                     bump = -1
                     N[j] = N[j] - 1
                     }
               tot = tot + N[j]
               }
P = P + bump
if (P <= 0) {
      P = P + 1
      while (N[P] = 0) P = P + 1
      }
if (tot != L) next (L)
if( P >= Nd ) STOP
      return()
}
```

[1] Source code in C for all three algorithms is available from the author.

Here the variables are defined:

```
N_d    Number of dimensions
N[j]   Vector containing the current point
nmax       maximum value allowed for each element of N[]
P      'level pointer'
L      Sum defining the current hyperplane
tot    The current total while generating point.
```

APPENDIX B: ALGORITHM FOR SHELL (L^*) SCANNING

This algorithm recursively descends to find all subspaces of the space to be scanned which are parallel to the original space. If the subspace has dimension zero, it is returned as a single point. If it has dimension one, it is iteratively enumerated. If it has dimension greater than one, it is divided into smaller subspaces. This selection is accomplished by binary counting. As a variable `jl` is incremented, its binary representation selects components of the space to form the new subspaces. The process is initialized with L equal to the maximum R-space value to be generated (in the examples, $L = 64$).

```
Sub Space Data Structure
      {
      int   dim;         /* dimensionality of SS */
      int   sub[NMAX];   /* list of selected dimensions */
      int   fill[NMAX]; /* constant values in non-selected dims */
      } Sub_Space;

RShell(L,S_S)
int L
Sub_Space S_S
{
int Ll,i,jl

if (dim(S_S) = 0) {
      eval(Map(S_S,-1))
      return(1)
      }
if(dim(S_S) = 1) {
      for(i=0;i<=L-1;i = i + 1)
            eval(Map(S_S,i))
      return(1)
      }
if(dim(S_S) > 1) {/* recursive sub scans of dimension 1 or 0 */
      Ll = L-1
      if(Ll < 0) return(1)
      for(jl=2^dim(S_S)-1; jl>=0; jl = jl - 1)/* generate sub spaces */
            RShell(Ll,Select(Ll,jl,S_S))/* scan the sub spaces */
      }
}
```

Here `Map(S_S,i)` is a function which returns a vector in which the elements specified by S_S are set to `i`, and others given the fill values. `Select(L,J,S_S)` (pseudocode below) is a function which generates a subspace structure which is the subspace of S_S which is selected by the binary representation of J. The dimensions which are not selected are given the value `L`. `''eval()''` is the function which is evaluated at the scanned point.

```
Select(L,J,S_S)
int   L,J
Sub_Space pointer S_S

{
int i,j
Sub_Space t

for(i=0;i<NMAX;i++) {
      t->fill[i] = S_S->fill[i]
      t->sub[i] = NULL_DIM
      }

j=0;

if(J & 1) t->sub[j++] = S_S->sub[0]
else t->fill[S_S->sub[0]] = L
if(J & 2) t->sub[j++] = S_S->sub[1]
else t->fill[S_S->sub[1]] = L
if(J & 4) t->sub[j++] = S_S->sub[2]
else t->fill[S_S->sub[2]] = L

...

if(J & 256) t->sub[j++] = S_S->sub[8];
else t->fill[S_S->sub[8]] = L;
if(J & 512) t->sub[j++] = S_S->sub[9];
else t->fill[S_S->sub[9]] = L;

t->dim = j;

return(t);

}
```

ACKNOWLEDGMENTS

The author acknowledges stimulating discussions with Eve Riskin, University of Washington Electrical Engineering Department, and Ross Baldick, Lawrence Berkeley Laboratory, valuable pointers to the mathematical literature from an anonymous referee, and the support of the University of Washington and the National Science Foundation Presidential Young Investigator Award.

REFERENCES

1. R. Baldick, personal communication, May 1991.
2. S. Deutsch, Pseudorandom dot scan television systems, *IEEE Trans. Broadcast* **BC-11,** July 1965, 11–21.
3. M. M. Gorbunov-Possadov, A. V. Ermakov, V. Ja. Karpov, M. P. Matekin, and I. M. Sobol, Applied program package CRIT for solving of multicriterial optimization problems in machine design, in *VTT Symposium 104, Software Development Trends, Joint Finnish–Soviet Software Symposium, Helsinki, Nov. 1988,* pp. 363–370.
4. H. Niederreiter, Quasi-Monte Carlo methods and pseudo-random numbers, *Bull. Amer. Math. Soc.* **64,** Nov. 1978, 957–1041.
5. K. N. Ngan, Image display using the cosine transform, *IEEE Trans. Acoust. Speech Signal Process.* **ASSP-32,** 1984, 173–177.
6. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing,* Cambridge Univ. Press, Cambridge, 1988.
7. K. Sloan and S. L. Tanimoto, Progressive refinement of raster images, *IEEE Trans. Comput.* **C-28** 1979, 871–874.
8. I. M. Sobol, Distribution of points in a cube and integration nets, *Uspehi Mat. Nauk* **21,** 1966. [In Russian]
9. I. M. Sobol, Quasi Monte Carlo methods, in *Proceedings, International Youth Workshop on Monte Carlo Methods and Parallel Algorithms, Promorsko, Bulgaria, Sept. 1989.* [Publ. 1991]
10. K. H. Tzou, Progressive image transmission: A review and comparison of techniques, *Opt. Engrg.* **26,** 1987, 581–589.