Exhaustive Geographic Search with Mobile Robots Along Space-Filling Curves[1]

Shannon V. Spires
Steven Y. Goldsmith

Advanced Information Systems Laboratory
Sandia National Laboratories
Albuquerque, New Mexico USA
svspire@sandia.gov, sygolds@sandia.gov

Abstract. Swarms of mobile robots can be tasked with searching a geographic region for targets of interest, such as buried land mines. We assume that the individual robots are equipped with sensors tuned to the targets of interest, that these sensors have limited range, and that the robots can communicate with one another to enable cooperation. How can a swarm of cooperating sensate robots efficiently search a given geographic region for targets in the absence of *a priori* information about the targets' locations? Many of the "obvious" approaches are inefficient or lack robustness. One efficient approach is to have the robots traverse a space-filling curve. For many geographic search applications, this method is energy-frugal, highly robust, and provides guaranteed coverage in a finite time that decreases as the reciprocal of the number of robots sharing the search task. Furthermore, it minimizes the amount of robot-to-robot communication needed for the robots to organize their movements. This report presents some preliminary results from applying the Hilbert space-filling curve to geographic search by mobile robots.

Introduction

The idea of using swarms of cooperating robots to solve various physical problems has received much attention recently [CFKM95], and has recently become cost-effective and practical with the advent of less expensive hardware and with new analysis techniques from complexity theory, chaos theory, and nonlinear dynamics. One problem that is particularly applicable to robot swarms is exhaustive geographic search. Exhaustive geographic search asks that we develop a complete map of all phenomena of interest within a defined geographic area, subject to the usual engineering constraints of efficiency, robustness, and accuracy [GR98b].

One example of such a search problem is that of finding buried land mines. It is possible to build a robot that has the requisite sensors and navigation apparatus to do this job, thus removing humans from an extremely high-risk activity. But a single robot is likely to be expensive and subject to damage from the mines, rendering it useless. If less expensive robots could be built in quantity, it would be better to build a swarm of such robots that could cooperate to locate all the mines within a given geographic region. With the right programming, many robots could do the job faster than one (ideally in $1/m$ the time, if m robots participate). The search process would be more robust because if a robot is damaged by a mine, others could take over its

[1] Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

DISCLAIMER

work.

This report presents an efficient, robust, cooperative search algorithm that allows robots to be applied to such a mission. The main idea of the algorithm is to have the robots traverse a minimum-length space-filling curve, dividing the search area among *m* robots.

Problem Constraints

The nature of the problem is such that there is no *a priori* knowledge of the locations of the targets (the targets are the land mines in the example above); we merely know that there *may* be one or more targets within a given geographic region. Our mission is to discover definitely their total number, their locations, and optionally to further characterize them along other dimensions. The requirements and constraints of the task are as follows:

- Each robot has a sensor apparatus which is adequate for detecting the targets of interest, but its range is limited to an area much smaller than the overall region of interest.
- Each robot can reliably communicate with the other robots with some probability P_c. The search mission should be completable even as P_c approaches 0, although it may take longer.
- Each robot has a finite energy supply (e.g. batteries or a fuel cell).
- The robots "know" when their job is finished.
- The robots do not collide.
- The robots stay within the bounds of the search region.
- A robot's individual search area should not overlap[2] that of another robot, but the entire search area must be covered (hence the use of the term *exhaustive* search).
- The search should still be completable even if one or more robots is damaged or otherwise fails during the mission. As long as at least one robot remains alive, the search should complete, although it may take longer.
- We must accomplish the search within a finite time.
- The search time should decrease as the number of robots participating increases.
- Control of the robots is completely decentralized; there is no central coordinating entity. The robots must be able to behave autonomously. Isolated robots must still be able to perform useful work.

[2] There may be cases where, for reasons of higher reliability, we explicitly *do* want them to overlap.

Geographic Search

In general, a geographic search will take place in three phases: (1) Initial configuration; (2) Search; (3) Terminal configuration [GR98a]. First, the robots must organize themselves into a coherent, communicating group within the region. They divide the area into subregions and assign each robot one or more subregions. Next, the search task proper is conducted. Finally, the robots determine through consensus that the region has been searched. At this point, the robots take some terminal action.

To maximize efficiency, two considerations are immediately obvious:

A) *Minimal configuration energy.* We don't want the robots to expend a great deal of energy in the initial configuration phase because no useful work is accomplished until the search phase. During initial configuration, the robots must expend energy communicating with each other and moving into their initial positions. These energy expenditures must be minimized.

B) *Optimal search coverage.* During the search phase, we must ensure that the entire search area is covered in a finite time, preferably proportional to the reciprocal of the number of robots involved.

To optimize (A), we could have the robots perform no initial configuration at all, with each simply searching the area in its immediate vicinity. No initial communication and no initial movement take place to coordinate with the other robots. We assume the robots are initially randomly distributed on the search region (because they were dropped out of an airplane, for example). Each robot could perform a random walk [W97a], spiral outward, or use some other autonomous algorithm. This defeats the purpose of multi-robot collaboration. Since there is no coordination, areas already searched can be retraced. Spiraling outward would guarantee that the search area was covered in a finite time, but using more than one robot would not necessarily decrease that time. A random walk is even worse—it wouldn't even guarantee coverage in a finite time.

Optimizing (B) means having the robots subdivide the search area equally and each search its agreed-upon subregion. A good deal of initial collaboration and movement would generally be needed here, especially if the robots start in random locations and must move to an organized initial configuration. An example of such an initial configuration is shown in Figure 1, where eight robots (triangles) have lined up in a column along the left side of the search region and plan to march to the right. Each search subregion is a horizontal strip.

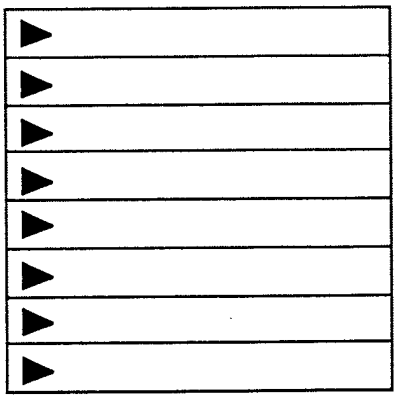

Figure 1: Robots (triangles) aligned in preparation to march to the right

This accomplishes a maximally-efficient search, but at the cost of a less efficient initial configuration. In addition, this search assumes all the robots are highly reliable; it's not very robust if one or more robots dies during the search. (We'll discuss why later in the report.)

The dilemma is thus to find the optimal tradeoff between the initial configuration phase and the search phase that makes both acceptably efficient, while ensuring the robustness of the mission.

We believe that searching along a space-filling curve solves the dilemma nicely: it requires minimal initial configuration while also guaranteeing a maximally-efficient, robust search. We will digress a bit here to describe space-filling curves in general.

Space-Filling Curves

A space-filling curve [S94] is a one-dimensional curve which passes through every point of a given N-dimensional region. Such curves can be constructed to fill regions of any dimensionality, but in our case we're mainly interested in two-dimensional regions, corresponding to a geographic area on the surface of the earth. Two-dimensional regions are also much easier to illustrate, and for those reasons the remainder of this report will concern only curves that fill 2-d regions. The reader should bear in mind, however, that the constructions and algorithms described herein can easily be extended to regions of more dimensions.[3]

Of course, a true space-filling curve is an ideal mathematical entity and can only be approximated in the real world. If we divide our region of interest into subregions of finite size, rather than points, we can easily draw a curve which passes through each subregion. In Figure 2, for example, a square region is divided into four subregions. The dark inverted U-shaped curve passes through the center of each subregion.

[3] A 3-d region [G97], [SLP83] might be appropriate for search by a swarm of undersea or flying robots, for example.

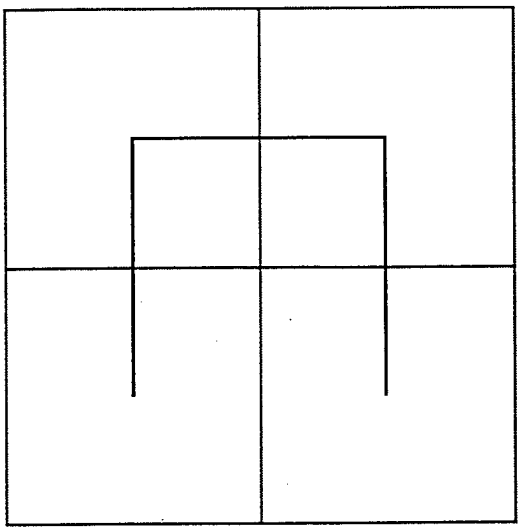

Fig. 2. First Order Hilbert curve

Subdividing further, into 16 regions (4 x 4) the curve becomes as in Figure 3.

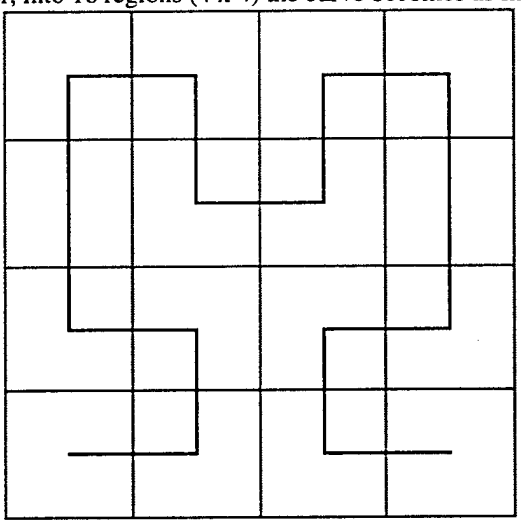

Fig. 3. Order 2 Hilbert curve

Subdividing still further, into an 8 x 8 grid, Figure 4 shows a curve which passes through each of the 64 subregions:

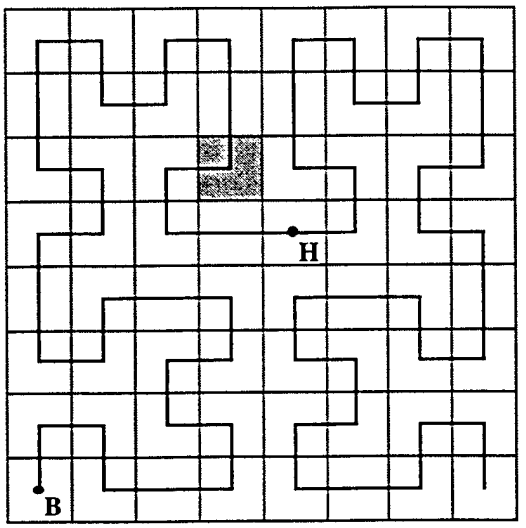

Fig. 4. Order 3 Hilbert curve

The family of curves illustrated in Figs 2-4 is called the *Hilbert* [W97b] curve after David Hilbert, their discoverer. There are many families of space-filling curves, but the Hilbert has some especially nice properties [MJFS96].

Notice how every deeper subdivision of the curve contains four copies of the entire previous curve, suitably rotated and reflected, with some straight segments added to ensure continuity. The curve is thus geometrically self-similar, as space filling curves frequently are. The *order* of the curve *D* determines the number of subregions it passes through and consequently, the total length of the curve:

$$\#\ \text{subregions} = (2^D)^2 = 2^{2D} = \text{Length}_{\text{curve}} + 1$$

Thus Figure 2 is a first order curve, or D=1, because it occupies 4 subregions and is 3 units long. Figure 3 has D=2, and Figure 4 has D=3. To fill every point in a region, a curve would have to be $D=\infty$.

The fact that the curve is a geometrically self-similar fractal implies that there is a recursive algorithm for its computation. This is indeed the case [McW97], [PLF91]. However, there is also a non-recursive algorithm for computing a given Hilbert curve which is more useful in our robotic swarm application.

If we wanted to identify a particular subregion of a Hilbert-traversed space—for example, the shaded subregion in Figure 4—we could do it in two different ways. We could specify its [x,y] location as [3,5] (where [0,0] specifies the lower left corner subregion marked with the letter B in the figure), or we could specify its position along the Hilbert curve as the scalar value 28, which indicates that it is the 28th subregion encountered along the curve measuring from the beginning of the curve at subregion B. Subregion B is location 0 in this frame of reference.

There is a straightforward algorithm to convert the 2-space coordinates of real space into a 1-space Hilbert coordinate and an equally simple algorithm to convert

from 1-space back into 2-space [B69]. Even though the dimensionality changes during this conversion, no information is lost because the total number of possible bits in x or y is D, while the total number of bits needed to specify the 1-space Hilbert coordinate is 2D. The conversion algorithms merely interpret the coordinate bits in different ways.

Search Along a Space-Filling Curve

By using 1-space to 2-space conversion algorithms, it is easy to program a robot to follow a space-filling curve. Assuming the robot is able to deduce its current x,y location (via Global Positioning Satellite, dead reckoning, etc.), the algorithm is shown in Table 1:

$x_0,y_0 \leftarrow$ current location
$H \leftarrow$ Convert-xy-to-Hilbert(x_0,y_0)
$H \leftarrow H + 1$
$x_1,y_1 \leftarrow$ Convert-Hilbert-to-xy(H)

Table 1: Hilbert traversal algorithm. x_1,y_1 represents the robot's new location.

In practice, we perform the operation $H \leftarrow H + 1$ modulo 2^{2D}, which causes the robot to return to the beginning point of the curve after it reaches the end; this closes the curve and makes it a topological circle instead of a line segment.

If we now give the robot a sensing mechanism whose sensory range is at least as large as a single subregion, by traversing the curve it can now search the overall region in time proportional to the number of subregions, 2^{2D}.

Notice that this algorithm works regardless of where on the curve the robot starts; it need not start at the H=0 point of the curve. This is one reason why the recursive algorithm is not used in the robotic application; it would be difficult to initialize the recursive control stack properly to allow traversal to begin at an arbitrary point. Another reason is that by repeatedly converting back and forth between real space and 1-space, correcting for real space navigation errors along the way is easy.

Starting at an arbitrary point becomes valuable when more than one robot is involved. If we have two robots, we could start one at the beginning of the curve (point B in Figure 4) and the other at the halfway point of the curve (point H in Figure 4); the resulting traversal of the curve takes place in half the time it would take with one robot. In general, if we start with m robots, we can search the space in time proportional to $2^{2D}/m$, provided we initially space the robots equally along the curve.

Efficiency

We'll now examine how search along a space-filling curve meets efficiency criterion (A). If we assume an initial random configuration of the robots within the region, we can actually get away with almost no initial configuration movement whatsoever. We

also assume that the search region is square[4] and that its bounds and the robots' individual sensor ranges are known by each robot *a priori.*

Configuration proceeds as follows:

1) Each robot senses its own initial location via some external mechanism (e.g. Global Positioning Satellite)

2) Each robot independently computes the subregion size based on its sensor range. (We assume for now that all the robots have the same sensor range, and thus compute the same subregion size.) Since each also knows the bounds of the search region, each also computes its own location relative to the search region and computes the order of the Hilbert curve based on the size of the subregions relative to the overall search region. Again, each robot will compute the same value for the order of the curve.

3) Each robot decides which subregion it is currently within by quantizing its x,y location relative to the overall region. It then moves to the center of that subregion. This is the only time its wheels need to turn during the initial configuration phase, and the maximum distance any robot will have to move is

$$\frac{s}{\sqrt{2}}$$

where s is the size of a subregion square measured along a side.

4) The robots broadcast their individual initial locations to each other so that each knows the starting locations of all the others. (How the robots reliably communicate this information is itself an interesting question but is beyond the scope of this report [G93].)

Configuration is now complete and the actual search begins. Each robot proceeds to each subregion in turn along the Hilbert curve, using the algorithm in Table 1. At each subregion, the robot senses any targets of interest and checks to see if its next subregion is the starting subregion of any other robot. If so, it stops and broadcasts "I'm Done!" along with information about any targets it may have found along the way. When all the robots have reported in, the collective can by consensus decide to report back to a higher authority, perform other tasks, or move on as a group to search another region. If after some reasonable time one or more robots have not reported in, the robot immediately behind the nonreporting robot can autonomously decide to proceed into the nonreporting robot's search area, search it, and report "I'm Done!". Note that each robot can precompute its own expected time to complete its portion of the search, since it knows at the outset how much of the curve it must traverse. These timeout values can be communicated to allow the collective to detect such fail-stop [K95] conditions.[5]

This leads us into an analysis of the expected search time. Certainly the time required to achieve guaranteed exhaustive coverage of the search region is bounded.

[4] Non-square regions and even general polygonal regions can be handled by extensions of the techniques presented here.

[5] The issue of Byzantine failures [K95], [LSP82], in which a robot pretends to cooperate but actually is a saboteur, must also be addressed in environments where malevolent robots may be present.

At worst, the search will take time proportional to the number of subregions in the search region (or equivalently, the length of the space-filling curve). The precise upper bound time, T_{UB}, can be calculated by multiplying the expected movement speed of a robot by s (the size of a subregion) by the total number of subregions. At best, the lower bound time will be $T_{LB} = T_{UB}/m$, with m the number of robots. The expected exhaustive coverage time T_E will equal T_{LB} when the robots are initially configured to be spaced equidistant from each other along the space-filling curve. But we're not doing that; we're just moving each to the center of the subregion it initially occupies.[6] In this kind of initial configuration, $T_E > T_{LB}$ but is still nowhere near T_{UB}. Empirical results based on Monte Carlo simulation are shown in Figure 5. The graph clearly shows a reciprocal relationship between T_E and number of robots. It shows that with 20 robots, the expected time to exhaust the space is less than 20% of the time required with 1 robot. This is obviously not as good as the theoretical best case of T_{LB} = 5%, but it's not bad for almost no initial configuration movement. We believe that with *slightly* more initial configuration movement, T_E can come much closer to T_{LB}. The nature of the particular space-filling curve being used becomes very important here. This is an open issue we are investigating.

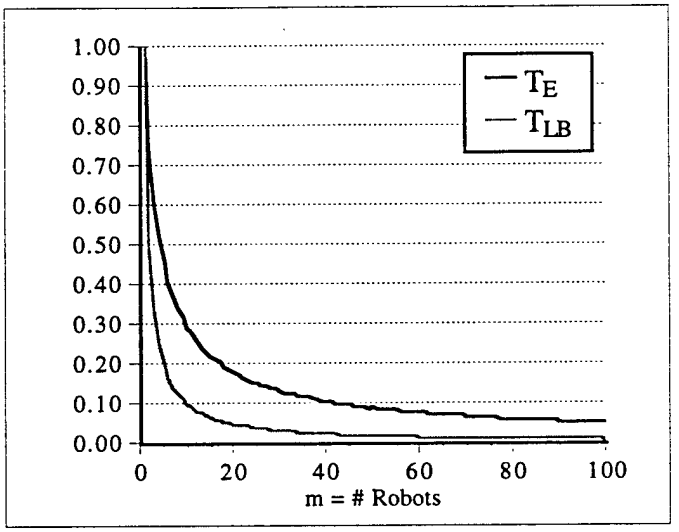

Fig. 5. T_E (upper curve) as a percent of T_{UB} vs. m. (T_{LB} is shown for comparison purposes.) Random initial positions and minimal initial configuration movement. Empirical results.

Robustness

In the initial configuration shown in Figure 1, where a space-filling curve is not being used, what happens if one of the robots is defective, or dies during the march? Its strip

[6] We assume for the sake of simplicity that no two robots initially occupy the same subregion. For a real implementation, a resolution protocol would be needed to deal with this possibility.

of territory will not be searched. The other robots will have to reliably detect this and reorganize themselves to search the defective robot's area. But if the robots are searching along a space-filling curve, all the robots are topologically searching along a single circle (Figure 6).

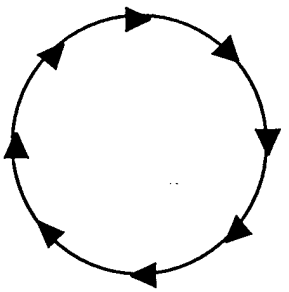

Fig. 6. Topological interpretation of search along a space-filling curve. Robots (triangles) traverse the curve clockwise.

Here, each robot eventually would search the entire space, because they are all searching along the same path. Ideally, each robot would stop when it encounters the starting point of the next robot along the path, with the result being that the space is searched in $1/m$ the time it would normally take with 1 robot. But if one robot breaks down, the next robot behind can simply continue along its path without stopping, and the overall space still gets searched, albeit in slightly longer time. No reconfiguration of the robots is necessary to accomplish this. Indeed, if as many as m-1 of the robots all break, the search is still guaranteed to complete (assuming the surviving robot has enough energy reserves to search the entire space).

The search is also robust with respect to communication breakdowns. If all the robots can communicate with each other, the only groupwide communications needed are for initial configuration, and when a robot encounters the starting point of the next robot along the path[7], it must announce to the collective "I'm Done!". When all robots have reported in, the collective knows the region has been completely searched. If one or more robots have not reported in by some time limit, the collective can deduce that they are dead and the next robot behind each dead robot can unilaterally decide (with no need for further communication with the group) to search the dead robot's area. If all communications fail (because of RF jamming, for example) the worst case is that the search will take as long as it would with only a single robot. Each robot will end up searching the entire space, and each will know to stop when it reaches *its own* starting point. The search will take a long time, but it's still guaranteed to complete.

Thus communications failures can cause the search mission to take longer, but they cannot prevent it from being accomplished.

[7] It can know where the next robot's starting point is either by memorizing the location during the initial configuration communication, or by having each robot drop a marker object on the ground at its own starting point. In the latter case, initial communication may be unnecessary.

Collision Avoidance

In any collaborative robotics system, collision detection and avoidance will be needed. Space-filling curves may make collision avoidance easier. When several robots follow a single space-filling curve, there is theoretically no chance of two robots' paths crossing, and therefore no need for dynamic collision avoidance. Realistically, two robots could collide if navigation errors are allowed to accumulate, if two adjacent robots along the curve travel at different speeds, or if the one in front dies. But it's possible that collision detection could be simplified because of the automatic separation inherent in the algorithm.

Further Research

Much work still needs to be done exploring the general problem of robot traversal of space-filling curves. Some avenues of exploration include formal expected exhaustion time analysis, handling robots with differing sensor ranges, non-square and non-rectangular search regions, automatic determination of the search region based on energy stores and initial positions of the extrema robots in a cluster, obstacle avoidance, and path planning. Space-filling curves other than the Hilbert need to be investigated as well. One disadvantage of the Hilbert curve is the long straight path that connects its endpoints when it is closed. Other curves have the property that their beginning and end points are adjacent. Also, in cases where there is a penalty for turns, other space-filling curves may do the job with fewer turns than the traditional Hilbert. The Hilbert II curve [W97b] might be a candidate here.

And of course, the tradeoff between minimal initial configuration energy vs. minimal search energy must be further explored in light of particular applications.

Conclusions

We have begun to explore how space-filling curves can enhance the efficiency and robustness of geographic search by robot collectives. Initial results have been quite promising, especially in applications such as mine-clearing where exhaustive search is necessary. Much work still needs to be done, but it appears that combining this kind of search with more information-exploitive search algorithms [GR98b] may be extremely useful in general real-world search situations.

We hope to have movies of the robot simulations, as well as an expanded version of this report, online soon at http://www.sandia.gov/aisl/robotics/.

References

1. [B69] Bially, T. Space-Filling Curves: Their Generation and Their Application to Bandwidth Reduction. *IEEE Transactions on Information Theory,* V 15 No. 6, November 1969, pp. 658-664.
2. [CFKM95] Cao, U., Fukunaga, A., Kahng, A., and Meng, F. 1995. Cooperative mobile robotics: Antecedents and directions. *Proc. of IEEE/RSJ IROS,* pp. 226-234.
3. [G93] Gage, D., How to communicate to zillions of robots. *Mobile Robots VIII, SPIE*, 250-257, 1993.
4. [G97] Gilbert, W., *A Cube-Filling Hilbert Curve,* http://math.uwaterloo.ca/~wgilbert/Research/HilbertCurve/HilbertCurve.html
5. [GR98a] Goldsmith, S., and Robinett, R. *Collaborative Search by Mobile Robots, Part I: Problem Definition.* Sandia National Laboratories Technical Report # . Preprint at http://www.sandia.gov/aisl/robotics/collaboration.
6. [GR98b] Goldsmith, S., and Robinett, R., *Collective Search by Mobile Robots Using Alpha-Beta Coordination.* Submitted to Collective Robotics Workshop, Agent World '98.
7. [K95] Kesteloot, L. *Fault-Tolerant Distributed Consensus.* http://tofu.alt.net/~lk/290.paper/290.paper.html
8. [LSP82] Lamport, L., Shostak, R., and Pease, M. The Byzantine Generals Problem, *ACM Transactions on Programming Languages and Systems 4*, 3 (July 1982), 382--401.
9. [MJFS96] Moon, B., Jagadish, H. V., Faloutsos, C. and Saltz, J. *Analysis of the Clustering Properties of Hilbert Space-filling Curve,* 1996 University of Maryland Technical Report CS-TR-3611, http://www.cs.umd.edu/TR/UMCP-CSD:CS-TR-3611
10. [McW97] McWhorter, W., *Fractint L-SystemVariations*, http://spanky.triumf.ca/www/fractint/lsys/variations.html
11. [PLF91] P. Prusinkiewicz, A. Lindenmayer and Fracchia, F. D., Synthesis of Space-filling Curves on the Square Grid. *Fractals in the Fundamental and Applied Sciences,* edited by Peitgen, H.-O. et al., Elsevier Science Publishers, 1991.
12. [SLP83] R. J. Stevens, A. F. Lehar, and F. H. Perston, Manipulation and presentation of multidimensional image data using the Peano scan. *IEEE Trans. on Pattern Analysis and Machine Intelligence,* PAMI-5(5) (1983) pp. 520-526.
13. [S94] Sagan, H. *Space-filling Curves*, Springer-Verlag, New York 1994.
14. [W97a] Weisstein, E., *Random Walk: 2-D,* http://www.astro.virginia.edu/~eww6n/math/RandomWalk:2-D.html
15. [W97b] Weisstein, E., *Hilbert Curve,* http://www.astro.virginia.edu/~eww6n/math/HilbertCurve.html

Report Number (14) SAND--98-0743C
CONF-980714--

Publ. Date (11) 199803
Sponsor Code (18) DOE/MA, XF
JC Category (19) UC-900, DOE/ER

DOE