Lattice Computers for Optimally Representing Euclidean Space

John Case Dayanand S. Rajan Anil M. Shende*

Department of Computer and Information Sciences,

103 Smith Hall, University of Delaware,Newark, DE 19716, USA.

1 Introduction

This TR expands and corrects slightly [CRS90a].

Existing applications for the prediction of the qualitative behavior of a physical phenomenon use analytical models of the phenomenon in computing the state of the phenomenon after a pre-determined time interval. These models involve sets of equations. (See [AG89].) The attributes of the objects involved (location in space, mass, velocity, acceleration, etc.) are the parameter values for the set of equations in an analytic model for the phenomenon and the methods for solving this set of equations are the procedures. Most *parallel* computer approaches concentrate on ingeniously parallelizing, in a manner suited to particular parallel architectures, the methods for solving the set of equations, and, then distributing data amongst the processors. Most often, this widens the gap between the problem being solved and its representation. In the case of physical phenomena, such approaches also neglect to use the inherent parallelism in the phenomenon (As a real object moves in real space, its component particles move in *parallel*. It is just this *natural* parallelism our approach, which we sketch in the next paragraph, exploits.). In particular, in these cases, the simulations do *not* unfold linear in the real–time for the corresponding phenomena to unfold.

We are interested in the *literal, analogical* simulation of physical phenomena using a mesh connected computer. We would like to model physical spaces discretely by mesh(-like) massively parallel computers (with local connectivity only). Here's the point of that. Each processor in such a mesh represents a point in space and each path built of a succession of immediate connections in the mesh represents a path in the space. Physical motion of tangible or intangible "objects" can be simulated by algorithmically passing, in the mesh, *fragments* of the objects from simulated point to adjacent simulated point in much the same way as the actual fragments of objects passing from point to point in real spaces constitutes the real motion of those objects. In a mesh those fragments could/would just be the algorithms governing how they are to be passed around. We require, of course, that our individual processors be synchronized. Clearly, then, our simulations can possibly be linear in real–time. In [She91, CRS91c, CRS91b] we present algorithms showing how one can achieve uniform motion in spite of the anisotropies *inherent* in any mesh–like representation of space.

*Research supported in part by NSF Grant CCR 8713846 at the University of Delaware. The email address for communication regarding this paper is 'case@cis.udel.edu'.

Our approach is similar, but *not equivalent to*, computational models of physics based on cellular automata [Fey82, Min82, FT82, Tof84, TM87, Sny47, Ung58, Mar84, SW86, Mac86, Tof77b, Tof77a, Vic84, Wol83, Zus69, Hil55, Svo86]. Lattice Gas models [Has87] are often studied using cellular automata to simulate complex physical phenomena. In a cellular automaton implementation of a Lattice Gas model, emphasis is placed on the simplicity of the local transition rule for the cellular automaton [FHP86]. Results are sometimes computed by the statistical averaging of the states over cells in pre-determined "neighborhoods" in the cellular automaton [Has87]. Our methodology differs from Lattice Gas models in both these regards. We do not restrict ourselves to extremely simple cellular automaton-like rules. Unlike individual cells in a cellular automaton, the individual processors in our meshes are computationally better equipped. In the introductory section of [CRS91b] we briefly discuss our approach to lattice gas problems; virtual directions, not averages are central.

Clearly, the accuracy of our approach depends on (i) the choice of a discrete representation of physical space, and (ii) the timing of each of the algorithms representing the local behavior of the "objects" in the phenomenon being simulated. Of course, since the algorithms must be "spatial" in nature, they will depend on the particular choice of a discrete representation of physical space.

In *this* paper, we deal only with the issue of discretely representing euclidean space *well*. Researchers in discrete physics who use cellular automata have noticed that some problems in physics need the dimension of the cellular automaton to be greater than 3 [Hil55, Tof77b, Svo86, Fey82]. Furthermore, [Tof77b] shows that every cellular automaton, C, can be made reversible by embedding C in a cellular space that has dimension one more than that of C. This is useful in modeling macroscopic phenomena [Tof77b]. So, we deal with the general issue of discretely representing euclidean n-space and do not restrict ourselves to euclidean 3-space.

Let $\mathbf{N} = \{0, 1, 2, \ldots\}$, the set of natural numbers. We consider interconnection schemes, for each dimension $n \in \mathbf{N}$, suitable for a mesh designed especially to simulate phenomena in euclidean n-space. We present a particular family, $\{\mathcal{M}(\mathbf{A}_n)\}_{n \in \mathbf{N}}$, of interconnection schemes and *prove that this family is optimal* for our purpose, under some reasonable, practical assumptions which we detail. Our assumptions on the interconnection schemes nicely combine criteria that the lattice–works resulting from the interconnection schemes model euclidean spaces well, and that the most basic, relevant operations can be carried out efficiently and cleanly. Interestingly, the family of underlying sets of points (defined in Section 3) $\{\mathbf{A}_n\}_{n \in \mathbf{N}}$ is well known, but no member of $\{\mathcal{M}(\mathbf{A}_n)\}_{n > 2}$ has (to our knowledge) been used as the hard-wired interconnection scheme for a mesh computer.

2 Euclidean n-Space and Literal Simulations

In this paper, $\mathbf{N}$, $\mathbf{Z}$, $\mathbf{Q}$, and $\mathbf{R}$ will denote the sets of natural numbers, integers, rationals, and reals, respectively. For $n \in \mathbf{N}$ and $R \in \{\mathbf{N}, \mathbf{Z}, \mathbf{Q}, \mathbf{R}\}$, R^n is the n-fold cartesian product $R \times \cdots \times R$. The (standard) orthonormal basis (see, e.g., [Her64]) for $\mathbf{R}^n$ is denoted by $\{\vec{e}_1, \ldots, \vec{e}_n\}$.

n-dimensional Euclidean space ($\mathbf{R}^n$) is continuous and infinite, and any (buildable) mesh (whose processors represent points in space) approximates at best only a bounded subset of it, by a *finite* number of "chosen" points. In order to consider possible candidates for a mesh that will be most suitable, it is best to remove the constraint of finiteness in the beginning and consider a countable (infinite) subset of $\mathbf{R}^n$ which approximates it and then intersect this approximation with a bounded subset of $\mathbf{R}^n$ in order to get one of our meshes.

Our rule for interconnecting processors (points) is that each processor is connected to all and only its nearest (measured by euclidean distance in $\mathbf{R}^n$) neighbors.

Each processor contains information regarding what kind of point it represents, e.g., whether the point is part of space occupied by an object or not. For instance, if a point is part of space that an object occupies, the corresponding processor will have information about the local properties of that object. This suggests that we will need a large number of processors in our mesh, *but* that each processor needs to be capable of storing only limited information and have limited computing ability. As pointed out earlier, we will, nonetheless, not restrict ourselves to the simplicity of the local transition rules of cellular automata.

The way we simulate motion, collision, forces etc. in our mesh is by propagating messages. Each processor will need to compute what its next state should be and then do accordingly. This can be done without a global monitor or controller provided each processor can store *small* programs which tell the processor what to do *in its present state*; then after computing its next state, the processor receives all relevant information (perhaps including programs) stored in a neighboring processor. This way, a point that "belonged" to an object in space can pass on relevant information to one of its neighbors and the object can "move" through the mesh. The object could be intangible, e.g., a wavefront, force, etc. In effect, the mesh as a whole, "pictures" the movement of the object.

Since motion is simulated by passing messages between processors, and all messages must be passed simultaneously, it is important (in the idealized, infinite case) that the local environment of each processor be pictorially similar to that of any other processor in the mesh. Hence, the relative locations of processors, as seen from any processor in the mesh, would be the same. This, in turn, implies that the *length* of each connection must be representative of the actual distance between processors, i.e., the time taken to transfer the same message across any connection in the mesh should be a constant.

In the next section, we formally describe the desirable properties mentioned in this section. We also show that there is an essentially unique approximation of euclidean n-space, which approximation is optimal for our purposes.

3 $\mathbf{A}_n$: An Optimal Approximation of Euclidean n-Space

Let D be a set of points in Euclidean n-space. We are interested in the possibility that points in D could correspond to processors in some *idealized* mesh computer (an infinite network of processors). Such an idealized mesh computer will be denoted by $\mathcal{M}(D)$ and will be referred to as the *idealized mesh computer based on D*. D is called the *underlying set* of $\mathcal{M}(D)$. More generally, given any subset X of D, $\mathcal{M}(X)$ will denote the sub-mesh of $\mathcal{M}(D)$ consisting of all those processors of $\mathcal{M}(D)$ corresponding to points in X, together with all possible interconnections which exist in $\mathcal{M}(D)$, between these processors. X is then the underlying set of $\mathcal{M}(X)$. In this paper, we will not make any distinction between the interconnection scheme for $\mathcal{M}(D)$ and $\mathcal{M}(D)$ itself. We introduce desirable properties (1 through 5) for D to have, in sequence.

Property 1 ALL OF EUCLIDEAN SPACE IS WELL REPRESENTED BY D; that is, we want D to be an "approximation" in the sense defined below.

Definition 1 *An* approximation *of* $\mathbf{R}^n$ *is a countable subset,* D*, of* $\mathbf{R}^n$ *so that there is a* $\mu = \mu(D) \geq 0$ *so that every open ball, in* $\mathbf{R}^n$*, with radius greater than* μ *contains at least one point of* D*.* μ *is then the* measure of proximity *of* D*.*

Informally, every point in Euclidean n-space is "not very far away" from D.

The sets $\mathbf{Q}^n$, $\mathbf{Z}^n$, $D_1 = \mathbf{Z}^n \setminus (2\mathbf{Z})^n$ and $D_2 = \mathbf{Z}^n \cup \{(x_1, \ldots, x_n) \mid x_j = (2k + 1/i), k, i \in \mathbf{Z}, i > 0\}$ are all approximations of $\mathbf{R}^n$, while $\{(\pm 2^{i_1}, \ldots, \pm 2^{i_n}) \mid i_1, \ldots, i_n \in \mathbf{Z}\}$ is not.

Definition 2 *An approximation* D *of* $\mathbf{R}^n$ *is* equivalent *to an approximation* D' *of* $\mathbf{R}^n$*, denoted by* $D \cong D'$*, iff there is a positive real number* m*, and an isometry (i.e., an isomorphism which preserves actual euclidean distances between corresponding points,)* $T : \mathbf{R}^n \to \mathbf{R}^n$ *so that* $m \cdot T(D) = D'$*.*

Note that $\cong$ is in fact an equivalence. Approximations belonging to the same equivalence class are similar in their "layout". For example, $\mathbf{Z}^2$ and $2\mathbf{Z}^2$ are equivalent.

Property 2 NO BOUNDED REGION OF $\mathbf{R}^n$ CONTAINS AN INFINITE SUBSET OF D.

We require this property because any one of our actual (buildable) meshes will be an intersection of a suitable bounded region of $\mathbf{R}^n$ with D. This is equivalent to the requirement that D be a "discretization":

Definition 3 *An approximation* D *of* $\mathbf{R}^n$ *is a* discretization *iff each point of* D *can be contained in an open ball, of* $\mathbf{R}^n$*, containing no other point of* D*.*

Of the four approximations in the examples above, $\mathbf{Q}^n$ and D_2 are not discretizations, while $\mathbf{Z}^n$ and D_1 are.

Physical connections between processors in the mesh cannot be of arbitrarily small length. So we require D to have the following property.

Property 3 POINTS IN D DO NOT CLUSTER TOO CLOSELY; that is, there is a real number, $\delta > 0$, so that the distance between any two points in D is at least δ. If there are two points in D so that the distance between these two points *is* δ, then we say D *has minimal distance* δ (or that the *minimal distance of* D *is* δ).

Obviously, if D has Property 3, then D has Property 2. The converse is not true, since, for example, the discretization $D = \mathbf{Z} \cup \{i + (1/i) \mid i \in \mathbf{Z}, i \neq 0\}$, of $\mathbf{R}$, has Property 2, but does not have Property 3.

Property 4 D IS INVARIANT UNDER TRANSLATIONS.

The following theorem is a combination of Theorems 1 and 2 proved in [GL87, pp. 18–19]. An *n-dimensional lattice* is an abelian subgroup (i.e., a $\mathbf{Z}$-submodule), of rank n, of $\mathbf{R}^n$. Intuitively, an n-dimensional lattice is a subset of Euclidean n-space, which subset is just like an n-dimensional vector-space, except that the scalar multipliers of vectors are restricted to be integers (for a precise treatment of lattices and their properties, again see [GL87]). The notions of vector, origin, basis, linear independence, linear transformation, isomorphism, etc., are all similar to those in Euclidean space (considered as a vector-space over the field of real numbers). The length of a vector in a lattice is the same as the length of that vector in Euclidean space.

Theorem 1 *A subset D of $\mathbf{R}^n$ is an n-dimensional lattice iff D has properties 1, 3 and 4.* □

Note that a translation invariant subset of $\mathbf{R}^n$, has Property 2 iff it has Property 3, so we have not lost Property 2.

Convention: From now on we will assume (without loss of generality) that, unless we explicitly say otherwise, *every lattice we mention has minimal distance 1.* Further, *D will always denote an n-dimensional lattice with minimal distance 1.*

The distance between points, P and Q, in a lattice is the length of the vector $(Q - P)$, from P to Q. So the minimal length of vectors in a lattice is exactly the minimal distance of that lattice considered as an approximation of euclidean space (of appropriate dimension). Henceforth, we will not distinguish between the points in a lattice and the corresponding vectors.

Definition 4 *The* set of directions *in D, denoted $Dir(D)$, is the set of all vectors, in D, of length 1.*

Lattices exist in abundance, and are quite varied in their properties [GL87, CN88]. Although the study of lattices is quite incomplete [GL87, CN88], it is known that a lot of facts, intuitively clear about $\mathbf{R}^n$, are not true in all lattices [GL87, CN88]. For example, there is *only one* (up to isometry) n-dimensional lattice, $\mathbf{Z}^n$, with an orthonormal basis. The presence of an orthonormal basis facilitates clean computation within $\mathbf{Z}^n$. Cleanness in computation is crucial for the efficient realization of our methodology in a mesh. Yet, considering the way we expect to use a mesh to simulate physical phenomena, it would be to our advantage if there were "many connections per processor". The reason is that we wish to represent as many local angles exactly as possible.

$\mathbf{Z}^n$ does not have the maximum number of direction vectors (among lattices with minimal distance 1). Hence, we will drop the requirement of having an orthonormal basis in the hope that we will find a lattice with more direction vectors, and in which computations are nearly as clean. To this end, we characterize $\mathbf{Z}^n$ using the property of regularity:

Definition 5 *A basis, $B = \{\vec{b}_1, \ldots, \vec{b}_n\}$, for D, is* regular *iff there is a constant, $K(D, B)$, so that the inner product*

$$(\vec{b}_i, \vec{b}_j) = \begin{cases} K(D, B), & \text{if } i \neq j, \\ 1, & \text{if } i = j. \end{cases}$$

D itself is said to be regular *iff D has a regular basis.*

Hence, a regular basis B is one in which the vectors in B have unit length and the angle between any two distinct vectors in B is a constant.

Having a regular basis is a powerful symmetry condition to the effect that with respect to some basis one dimension is not to be preferred to another. The existence of a regular basis is important for both computational cleanness and having a lattice that *nicely* represents the corresponding euclidean space of the same dimension. Our methodology dictates that our algorithms [She91, CRS91c, CRS91b] be local and independent of any fixed co-ordinatization of euclidean space. Vector inner product is an *extremely* basic operation in our algorithms; having its calculation simplified is guaranteed by the existence of a regular basis. This makes computations more efficient.

The following characterization of $\mathbf{Z}^n$ shows that it is regular in the strongest possible sense.

Proposition 1 *$\mathbf{Z}^n$ is the unique (up to isometry) regular lattice, D, in which* every *basis contained in $Dir(D)$ is regular.* $\square$

On the other hand, there are numerous lattices, D, satisfying

Property 5 D IS REGULAR.

The best one can hope for is that there is a unique (up to isometry) regular n-dimensional lattice (on which to base an idealized mesh) for which the number of direction vectors is maximum. We now state and prove a theorem (Theorem 2) which, by explicitly giving expressions for the number of direction vectors in regular lattices, shows us that our hope can be fulfilled (see Theorem 3). The corollaries that follow isolate some other useful consequences.

The following lemma is easy to prove, and helps show that Theorem 2, in fact, covers all regular lattices.

Lemma 1 *In D (*recall the convention above that D has minimal distance 1*) the inner product of any two non-collinear vectors $\vec{d}_1, \vec{d}_2$ in $Dir(D)$ lies in the interval $[-1/2, 1/2] \subset \mathbf{R}$.* $\square$

Theorem 2 *Let $n > 1$, and let D be regular with regular basis B, and, let $k = K(D, B)$. Then,*

$$card(Dir(D)) = \begin{cases} 2n, & \text{if } -\frac{1}{n} < k < \frac{1}{2}; \\ n(n+1), & \text{if } k = \frac{1}{2}; \\ 2(n+1), & \text{if } k = -\frac{1}{n}. \end{cases}$$

PROOF. Let $\vec{d} = \sum_{i=1}^{n} p_i \vec{b}_i \in Dir(D)$, $\vec{b}_i \in B, p_i \in \mathbf{Z}$. Then $1 = \| \vec{d} \|^2 = \sum_{i=1}^{n} p_i^2 + 2k \sum_{j>i=1}^{n} p_i p_j$.
CASE 0: $2k \sum_{j>i=1}^{n} p_i p_j = 0$. Then $1 = \sum_{i=1}^{n} p_i^2$. So $\vec{d} = \pm \vec{b}_i$ for some $\vec{b}_i$, giving us $card(Dir(D)) = 2n$ in this case.
CASE 1: $0 \le k < 1/2$.
This case reduces to Case 0: If $k = 0$, this is obvious, so suppose $k > 0$. Then,
$\sum_{i=1}^{n} p_i^2 + 2k \sum_{j>i=1}^{n} p_i p_j = 1$, so, $\sum\limits_{j>i=1}^{n} p_i p_j \le 0$. If $\sum_{j>i=1}^{n} p_i p_j \ne 0$, then,

$$1 > \sum_{i=1}^{n} p_i^2 + \sum_{j>i=1}^{n} p_i p_j > \sum_{i=1}^{n} p_i^2 + 2 \sum_{j>i=1}^{n} p_i p_j \ge 0.$$

Since each p_i is an integer, $\sum_{i=1}^n p_i^2 + 2\sum_{j>i=1}^n p_i p_j = 0$, and, $\sum_{i=1}^n p_i^2 + \sum_{j>i=1}^n p_i p_j = -\sum_{j>i=1}^n p_i p_j = 0$, giving us a contradiction. Therefore, $card(Dir(D)) = 2n$ in this case.

Case 2: $k = 1/2$. Then, $1 = \sum_{i=1}^n p_i^2 + \sum_{j>i=1}^n p_i p_j$. Therefore, $0 \geq \sum_{j>i=1}^n p_i p_j$, which gives

$$1 \geq 1 + \sum_{j>i=1}^{n} p_i p_j = \sum_{i=1}^{n} p_i^2 + 2 \sum_{j>i=1}^{n} p_i p_j = \left(\sum_{i=1}^{n} p_i \right)^2 \geq 0.$$

So, $\sum_{j>i=1}^n p_i p_j = 0 \; or \; -1$.

If $\sum_{j>i=1}^n p_i p_j = 0$ then Case 0 applies, and we have the $2n$ possibilities: $\vec{d} = \pm \vec{b}_i$. Otherwise, $\sum_{j>i=1}^n p_i p_j = -1$, and $\sum_{i=1}^n p_i^2 = 2$ and so, $\sum_{i=1}^n p_i = 0$. Thus, exactly two of the p_i's are non-zero; one of the non-zero p_i's is a 1 and the other is a -1. This gives us $n(n-1)$ choices for $\vec{d}$. So, in all we have $n(n-1) + 2n = n(n+1)$ choices for $\vec{d}$ in this case, i.e., $card(Dir(D)) = n(n+1)$.

Case 3: $-1/2 \leq k < 0$.

Claim 1: For each $i, p_i \geq 0$ or for each $i, p_i \leq 0$.

Let $p_{i_0} \neq 0$, $1 \leq i_0 \leq n$. Then, $1 = \sum_{i=1}^n p_i^2 + 2k\sum_{j>i=1}^n p_i p_j = \| \vec{d} - p_{i_0} \vec{b_{i_0}} \|^2 + p_{i_0}^2 + 2kp_{i_0}(\sum_{i=1,\ i \neq i_0}^n p_i)$. Therefore, $-2kp_{i_0}(\sum_{i=1,\ i \neq i_0}^n p_i) = \| \vec{d} - p_{i_0} \vec{b_{i_0}} \|^2 + p_{i_0}^2 - 1$. Since, $-2k > 0$, we add $p_{i_0}^2$ to both sides above to get $p_{i_0}(\sum_{i=1}^n p_i) \geq p_{i_0}^2 > 0$, thus showing that Claim 1 is true.

Claim 2: $k = -1/n$.

To show that $k \leq -1/n$, it is enough to prove that $n(\sum_{i=1}^n p_i^2 - 1) - 2\sum_{i,j=1}^n p_i p_j \geq 0$, because, $k = -(\sum_{i=1}^n p_i^2 - 1)/(2\sum_{i,j=1}^n p_i p_j)$ and $\sum_{i,j=1}^n p_i p_j > 0$. But,

$$n(\sum_{i=1}^{n} p_i^2 - 1) - 2 \sum_{i,j=1}^{n} p_i p_j = \sum_{j>i=1}^{n} (p_i^2 - 2p_i p_j + p_j^2) + \sum_{i=1}^{n} p_i^2 - n = \sum_{j>i=1}^{n} (p_i - p_j)^2 + \sum_{i=1}^{n} p_i^2 - n.$$

Now, if $s \geq 1$ of these p_i's are non-zero, then $\sum_{i=1}^n p_i^2 \geq s$ and $\sum_{j>i=1}^n (p_i - p_j)^2 \geq (n-s)s$. So,

$$\sum_{j>i=1}^{n} (p_i - p_j)^2 + \sum_{i=1}^{n} p_i^2 - n \geq s + (n-s)s - n = (n-s)(s-1) \geq 0.$$

Thus, we have $k \leq -1/n$. To prove that $n - 1 \geq -kn(n-1)$, let $\vec{v}$ be the vector $\sum_{i=1}^n \vec{b}_i$. Then,

$$\| \vec{v} \|^2 = \sum_{i=1}^{n} 1^2 + 2k \sum_{j>i=1}^{n} 1 \cdot 1 = n + 2k \frac{n(n-1)}{2} = n + kn(n-1) \geq 1.$$

So,$k \geq -1/n$, and Claim 2 is proved.

Claim 3: Let, without loss of generality, $p_i > 0$ for $1 \leq i \leq m \leq n$ and $p_i = 0$ otherwise. Then, $m = n$ or $m = 1$ and $p_i = 1$ for $i \leq m$.

Let $p_i = 1 + q_i$, $q_i \geq 0$ for $1 \leq i \leq m$ and let $\vec{w} = \sum_{i=1}^m q_i \vec{b}_i$. Then,

$$\begin{aligned} \| \vec{d} \| &= \sum_{i=1}^{m} p_i^2 - \frac{2}{n} \sum_{j>i=1}^{m} p_i p_j = \sum_{i=1}^{m} (1+q_i)^2 - \frac{2}{n} \sum_{j>i=1}^{m} (1+q_i)(1+q_j) \\ &= m - \frac{m(m-1)}{n} + \| \vec{w} \|^2 + (2 - \frac{2(m-1)}{n})(\sum_{i=1}^{m} q_i). \end{aligned}$$

Now, $m - (m(m-1)/n) \geq 1$ for $1 \leq m \leq n$, and equality holds iff $m = 1$ or $m = n$. Also, the last two terms are positive unless every q_i is 0, i.e., unless every $p_i = 1$. Thus, Claim 3 is true.

Thus, in the case $-1/2 \le k < 0$, we get, $\| \vec{d} \| = 1$ iff $\vec{d} = \pm\vec{b}_i$ or $\vec{d} = \pm\sum_{i=1}^{n} \vec{b}_i$. Therefore, $card(Dir(D)) = 2n + 2 = 2(n+1)$. $\square$

The following three corollaries follow from the proof of the above theorem.

Corollary 1 *For an n-dimensional regular lattice, D, with regular basis, B, $-1/n \le K(D,B) \le 1/2$.* $\square$

Corollary 2 *Let D be an n-dimensional regular lattice with regular basis, B.*
Then, $Dir(D) = P(B) \cup Q(B)$, where,

$$
\begin{array}{rcl}
P(B) & = & \{ \pm\vec{b} \mid \vec{b} \in B \} \ \ \textit{and} \\
Q(B) & = & \begin{cases} \emptyset & \textit{if } -1/n < K(D,B) < 1/2; \\ \{ (\vec{b} - \vec{b}') \mid \vec{b}, \vec{b}' \in B,\ \vec{b} \ne \vec{b}' \} & \textit{if } K(D,B) = 1/2; \\ \{ \pm(\sum_{\vec{b} \in B} \vec{b}) \} & \textit{if } K(D,B) = -1/n. \end{cases}
\end{array}
$$

Corollary 3 *For an n-dimensional, $n \ge 2$, regular lattice, D, with regular bases B, C,*

$$K(D,C) = \begin{cases} \pm K(D,B) & \textit{if } n = 2 \\ K(D,B) & \textit{otherwise.} \end{cases}$$

Proof. We first show that (i) $K(D,C) = \pm K(D,B)$ for every n, and then that (ii) $K(D,C) \ne -K(D,B)$ if $n > 2$.

To prove (i), it is enough to show that for some $\vec{c}_1, \vec{c}_2 \in C$, $\vec{c}_1 \ne \vec{c}_2$, $(\vec{c}_1, \vec{c}_2) = \pm K(D,B)$. $B, C \subset Dir(D)$. The previous corollary provides us with explicit descriptions of the possible forms for $\vec{c}_1$ and $\vec{c}_2$ in each of the three cases for $K(D,B)$.

Case 1: $0 \le K(D,B) < 1/2$. $C \subset P(B)$ (in the notation of the previous corollary). So, $\vec{c}_1 = \pm\vec{b}_1$ and $\vec{c}_2 = \pm\vec{b}_2$, for some $\vec{b}_1, \vec{b}_2 \in B$, and hence $(\vec{c}_1, \vec{c}_2) = \pm K(D,B)$.

Case 2: $K(D,B) = 1/2$. $card(Dir(D)) = n(n+1)$ in this case. Suppose $K(D,C) \ne 1/2$. Then, $-1/2 \le K(D,C) < 1/2$. If $0 \le K(D,C) < 1/2$, then, $card(Dir(D)) = 2n \ne n(n+1)$ giving us a contradiction. If $K(D,C) < 0$, then, $card(Dir(D)) = 2n + 2$, which is equal to $n(n+1)$ iff $n = 2$. Then, $K(D,C) = -1/n = -1/2 = -K(D,B)$.

Case 3: $K(D,B) = -1/n$. If $C \subset P(B)$, then the proof is identical to the proof of Case 1. If $\vec{c}_1 = \pm\vec{b}_1 \in P(B)$ and $\vec{c}_2 \in Q(B)$, then, $(\vec{c}_1, \vec{c}_2) = (\pm\vec{b}_1, \pm(\sum_{\vec{b} \in B} \vec{b})) = \pm(1 - (n-1)/n) = \pm 1/n = K(D,B)$. The case $\vec{c}_1, \vec{c}_2 \in Q(B)$ cannot arise since each element in $Q(B)$ is the additive inverse of the other, and hence $Q(B)$ is not a linearly independent set.

To prove (ii), notice that

Case 1: if $-1/n < K(D,B) < 1/2$ and $c_1, c_2, c_3 \in C \subset P(B)$, then it is not possible that $(c_i, c_j) = -K(D,B)$ for all $i \ne j$ with $1 \le i,j \le 3$.

Case 2: if $K(D,B) = 1/2$, then $K(D,C) = -K(D,B)$ is possible iff $n = 2$;

Case 3: if $K(D,B) = -K(D,C) = -1/n$, then $card(Dir(D))$ can be expressed simultaneously as $2n+2$ and also as $2n$ or $n(n+1)$. This is possible iff $n = 2$ and $card(Dir(D)) = n(n+1)$. $\square$

As mentioned earlier, our algorithms sometimes depend, in effect, on calculations of $K(D, B)$. By the last corollary, the quantity $K(D, B)$ does not depend on the choice of regular basis B, except possibly in dimension 2. Hence, the concept defined in the definition just below is well-defined; furthermore, the lack of dependence on regular basis is good, computationally, for our purposes, as we will see in Section 4.1.

Definition 6 *The* regular inner product *of an n-dimensional regular lattice, D, denoted $K(D)$, is defined by*

$$K(D) = \begin{cases} |K(D, B)| & \text{if } n = 2 \\ K(D, B) & \text{otherwise.} \end{cases}$$

for some regular basis B for D.

It is not difficult to construct, for each positive integer, $n > 2$, and for each k in the interval $[-1/n, 1/2]$ the unique (up to isometry) n-dimensional regular lattice with regular inner product equal to k. We denote by $\mathbf{A}_n$, the n-dimensional regular lattice whose regular inner product is $1/2$.

$\mathbf{A}_n$ is equivalent to the well known (see [CN88, GL87]) lattice, A_n, defined in Section 4.1. For each $n > 1$, the dual lattice, A_n^* (see [CN88, GL87]), of A_n, is equivalent to the unique (up to isometry) n-dimensional regular lattice with regular inner product equal to $-1/n$.

We conclude this section with the following important theorem (a direct consequence of Theorem 2) which asserts the existence and uniqueness of a mesh computer, whose underlying set satisfies Properties 1 through 5 (i.e., is a regular lattice), and which has maximum connectivity.

Theorem 3 *The (unique) idealized mesh computer, $\mathcal{M}(\mathbf{A}_n)$, has maximum connectivity among all idealized meshes based on n-dimensional regular lattices.* □

4 Some Additional Merits of $\mathbf{A}_n$

As stated above, the lattice A_n, which is equivalent to $\mathbf{A}_n$, is well known and a substantial number of its properties have been tabulated [CN88, GL87]. This is useful because $\mathcal{M}(A_n)$ has the same (network) topology as $\mathcal{M}(\mathbf{A}_n)$, and so, every property of A_n contributes to some property of $\mathcal{M}(\mathbf{A}_n)$. In the following sections (Sections 4.1 through 4.3), we state some of the elementary properties of $\mathbf{A}_n$, and mention their relevance to our approach of literally simulating physical phenomena in $\mathcal{M}(\mathbf{A}_n)$. Finally, in Section 4.4, we discuss why we do not choose to base our meshes on lattices that are denser than $\mathbf{A}_n$.

4.1 Properties of $\mathbf{A}_n$ Related to Computational Cleanness

One of the main reasons for using square meshes, is that a lot of the commonly used quantities, such as inner product, are easy to compute in the lattice $\mathbf{Z}^n$. The properties of $\mathbf{A}_n$ described in this section indicate that some of these computations are almost as easy in $\mathbf{A}_n$.

Ease of representation: The set $A_n = \{ (x_0, \ldots, x_n) \in \mathbf{Z}^{n+1} \mid \sum x_i = 0 \}$ is equivalent to $\mathbf{A}_n$ in $\mathbf{R}^{n+1}$.

In other words, there is an additive group isomorphism, from $\mathbf{A}_n$ to A_n, which isomorphism scales distances by a constant factor. A_n is necessarily an n-dimensional lattice contained in $\mathbf{R}^{n+1}$. In this representation, A_n, of $\mathbf{A}_n$, the constant scaling factor, and hence the minimal distance of A_n, is $\sqrt{2}$. This is a good representation for $\mathbf{A}_n$ as is indicated by the following features.

1. All points in A_n have integer co-ordinates, thus aiding in restricting ourselves to integer computations only (in fact, accurately representing the co-ordinates of points in $\mathbf{A}_n$, in terms of the standard basis for $\mathbf{R}^n$, is not possible without symbolic computation, due to the presence of irrationals).

2. Each direction vector in $\mathbf{A}_n$ is represented easily in A_n as the difference, $(\vec{e}_i - \vec{e}_j)$, of two distinct standard basis vectors of $\mathbf{Z}^{n+1}$.

3. An important consequence of 1 above is that the usual inner product of any pair of vectors in A_n is an integer. Thus, although the euclidean distances between points in A_n need not be integral, the squares of these distances are.

4. The hardware implementation of the computation of the usual inner product of two vectors in A_n can be easily parallelized.

5. The test for membership in A_n, of a point in $\mathbf{Z}^{n+1}$, is computationally very easy.

6. The path length, in A_n, between points P and Q in A_n is the sum of all the positive co-ordinates in the vector $(P - Q)$. This facilitates estimating the minimum time required for the propagation of a message from the processor at P to the processor at Q in $\mathcal{M}(\mathbf{A}_n)$.

7. Using this representation (of $\mathbf{A}_n$ by A_n), it is easy to derive expressions for certain relevant quantities. For example, it is easy to derive (and compute) the expression $\sum_{r=0}^{R} \sum_{i=0}^{n+1} (-1)^i \binom{n+1}{i} \binom{n+r-i}{n}^2$ for the number of processors at most R connections away from a given processor in $\mathcal{M}(\mathbf{A}_n)$.

Predecessor containment: *Every* $k < n$ dimensional sublattice of $\mathbf{A}_n$, spanned by k vectors from a regular basis B for $\mathbf{A}_n$, is isometric with $\mathbf{A}_k$.

A similar property is true for $\mathbf{Z}^n$. $\mathcal{M}(\mathbf{A}_n)$ contains several copies of every predecessor, $\mathcal{M}(\mathbf{A}_k), k < n$, of $\mathcal{M}(\mathbf{A}_n)$ in the family of idealized meshes $\mathcal{M}(\mathbf{A}_n)$. This allows our algorithms to be nicely parameterized by dimension.

Spanning set: *Every* set of n linearly independent, minimal length vectors in $\mathbf{A}_n$ generates (spans) $\mathbf{A}_n$.

There are lattices not having this property [GL87]. Suppose B is a set of n linearly independent, minimal length vectors, in such a lattice, such that B does not span that lattice. Intuitively, if one tries to walk about in that lattice using *only* the vectors in B as basic steps, there are points, actually in the lattice, that one will not be able to reach. Of course the lattice that *is* spanned by B will be n-dimensional too. Such lattices (not having the spanning set property) are *very* unlike n-dimensional euclidean space.

4.2 Comparison of $\mathbf{A}_n$ with $\mathbf{Z}^n$

Currently, most meshes in use [TM87, PD84, Uhr72, Dye, AS84, PF+85, Lev85, DL81, PU82] are essentially (though, some of these meshes make toroidal connections) two-dimensional square meshes, i.e., essentially meshes based on $\mathbf{Z}^2$. $\mathcal{M}(\mathbf{A}_n)$ is preferable over $\mathcal{M}(\mathbf{Z}^n)$ for several reasons:

Embedding: $\mathbf{Z}^n$ is *isometrically* embeddable in $\mathbf{A}_{2n-1}$, but $\mathbf{A}_n$ is not embeddable in $\mathbf{Z}^m$ for any m.

Density of points: For each n, $\mathcal{M}(\mathbf{A}_n)$ offers far better resolution, in the number of points of euclidean space approximated by the underlying set, than does $\mathcal{M}(\mathbf{Z}^n)$. In fact, we can show that given an n-dimensional ball, $\mathcal{B}_r$, of euclidean radius $r \in \mathbf{R}$, the ratio of the number of points of $\mathbf{A}_n$ in $\mathcal{B}_r$ to the number of points of $\mathbf{Z}^n$ in $\mathcal{B}_r$ is exponential in the dimension n.

Well covering of $\mathbf{R}^n$: Suppose δ is the length of minimal length vectors in D. For each d in D, informally we define the *sphere of influence* of d as the set of all points in euclidean n-space of distance $< \delta$ from d. It is clearly desirable that each point in euclidean n-space reside in the sphere of influence of some point in D, because, then, in a sense, euclidean n-space is *well-covered* by the points in D. Consider the measure of proximity μ, for D. We have that D well-covers euclidean n-space iff $\delta > \mu$. For $\mathbf{A}_n$, the lattice corresponding to our idealized n-dimensional mesh, the maximum dimension for which $\delta > \mu$ is 6, but for $\mathbf{Z}^n$, the lattice corresponding to the idealized n-dimensional square mesh, this maximum dimension is only 3. Further, for each n, $\mu(\mathbf{A}_n) < \mu(\mathbf{Z}^n)$. Therefore, the family $\{ \mathcal{M}(\mathbf{A}_n) \}$ is superior to the family, $\{ \mathcal{M}(\mathbf{Z}^n) \}$ of square meshes also in regard to well-covering.

Proximity of processors: The number of processors at most R connections away from a given processor in $\mathcal{M}(\mathbf{Z}^n)$ is given by $\sum_{r=0}^{R} \sum_{i=0}^{n}(-1)^i \binom{n}{i} \binom{n+r-i-1}{n-1}$. Comparing this with the corresponding expression for $\mathcal{M}(\mathbf{A}_n)$ from feature 7 in Section 4.1, we find that the number of processors at most R connections away is less in $\mathcal{M}(\mathbf{Z}^n)$ than in $\mathcal{M}(\mathbf{A}_n)$, for each $n > 1$. In particular, the number of near neighbor connections for $\mathcal{M}(\mathbf{A}_n)$ is $n(n+1)$, a feasible quadratic in n, *but*, for $\mathcal{M}(\mathbf{Z}^n)$, the number is only $2n$, a small linear expression in n.

Interconnection bandwidth across partitions: The predecessor containment property of $\mathbf{A}_n$ (see Section 4.1 above) ensures that $\mathbf{A}_n$ can be partitioned cleanly and naturally.

If $B = \{ \vec{b}_1, \ldots, \vec{b}_n \}$ is a regular basis for D, $P_D(t_1, \ldots, t_n) = \sum_{i=1}^{n} p_i \vec{b}_i$, $p_i \in \mathbf{Z}$, $0 \leq p_i < t_i$ is a *standard n-dimensional parallelepiped*, in D, with thickness t_i in direction $\vec{b}_i$ [CRS90b]. The submesh, $\mathcal{M}(P_D(t_1, \ldots, t_n))$, of $\mathcal{M}(D)$ is called a *parallelepiped mesh (with thickness t_i in direction $\vec{b}_i$)*. The predecessor containment property of D ensures that each face of a standard n-dimensional parallelepiped is the translate of a *standard* $(n-1)$ dimensional parallelepiped.

Let t be a positive integer, and let $P_D = P_D(t, \ldots, t) \subset D$. An $(n-1)$-dimensional plane, C, parallel to one of the $(n-1)$-dimensional faces of P_D, which plane does not pass through any processors of $\mathcal{M}(P_D)$, partitions $\mathcal{M}(P_D)$ into two components, each of which is (a translate of) a parallelepiped mesh.

The number, $B_D(t)$, of processor *interconnections*, of $\mathcal{M}(P_D)$, that pass through C is the *bandwidth of the partition (by C of $\mathcal{M}(P_D)$)*. We can show that $B_{\mathbf{A}_n}(t) = t^{n-2}(nt - n + 1)$ and $B_{\mathbf{Z}^n}(t) = t^{n-1}$, so that $\lim_{t\to\infty} B_{\mathbf{A}_n}(t)/B_{\mathbf{Z}^n}(t) = n$. Hence, in general, our reasonably thick parallelepiped meshes of dimension n will have about n times as many "parallel communication channels" from one partition to the other compared to an analogous n-dimensional square mesh. For our philosophy/methodology of computing, this is a good thing.

Automorphism groups: The order of the automorphism group of $\mathbf{A}_n$ is $2 \cdot (n+1)!$ while that of $\mathbf{Z}^n$ is $2^n \cdot n!$ [CN88]. The automorphism group is an important consideration for Lattice Gas theorists because of the simplicity of their local transition rules [Has87, FHP86]. It is not as important a consideration for us, although we note that the automorphism group of $\mathbf{A}_n$ is not of extremely low order.

Tessellating euclidean space: One common reason for using $\mathcal{M}(\mathbf{Z}^n)$ is that n-dimensional cubes are the only regular bodies that tessellate euclidean n-space, for $n > 2$ [CN88]. We do not wish to tessellate space with copies of a single regular body; *having a low measure of proximity for the underlying set is sufficient for our purposes.* As seen in Section 4.2, by this criterion, $\mathcal{M}(\mathbf{A}_n)$ is superior to $\mathcal{M}(\mathbf{Z}^n)$. Further, we would like to point out that the set of standard n-dimensional parallelepipeds, each with thickness 2 in each direction (in a regular basis) in $\mathbf{A}_n$, does, in fact, tessellate euclidean n-space. Of course, these parallelepipeds are not as symmetric as n-dimensional cubes.

4.3 Embedding $\mathcal{M}(\mathbf{A}_n)$ In Physical 3-Space

For *actually building* any finite sub-mesh of $\mathcal{M}(\mathbf{A}_n)$, of possibly high dimension, we are constrained by physical reality—a *physical instance* of any one of these finite sub-meshes, ostensibly, must be *physically embeddable* in euclidean 3-space. Furthermore, for our approach, the embedding in euclidean 3-space must be such that the physical length of the connection between adjacent processors is a *feasible constant*. Constant length interconnections are clearly important to help achieve *synchronous* motion of the "particles" making up a simulated object. For a given size of processor (of course > 0), we *can* embed $\mathcal{M}(\mathbf{A}_n)$ in physical 3-space with constant length interconnections. *In our embedding*, the minimal (constant) length of the interconnection between immediately connected processors does depend, but only *sub*polynomially, on the dimension of the mesh; furthermore, for each fixed dimension, it does *not* depend on the number of processors in the mesh. We will report elsewhere on the details of our physical embedding schemes.

4.4 Why Not Meshes Based on Denser Lattices?

Consider the classical problems of packing unit sized n-spheres into euclidean n-space (i) as densely as possible and (ii) so that each sphere touches the maximum number of neighbors [CN88, GL87]. Consider a solution to one (or both) of these problems. One could define a corresponding mesh (for approximating euclidean n-space) with processors at the centers of each sphere and its near neighbor connections to all and only those processors at the center of the spheres touching it. In general, the set D of points corresponding to the centers of the spheres need not even be a lattice [CN88, GL87]. Hence, such D would be too hard to deal with computationally and fail to have very important properties in common with euclidean n-space. A more severe consideration is that, for most cases of interest, the solutions to problems (i) and (ii) above are just unknown.

While more is known about problems (i) and (ii) *modified* to further require the solutions to be lattices, only a few cases of interest are solved [CN88], and the solutions, for dimensions ≥ 4, all *must* fail to be regular. Also, the (few) known solutions apparently come from totally different *families* of lattices [CN88], and, hence, it would be hard to design and *prove* algorithms for use in meshes based on such lattices.

5 Related Work

As indicated in Section 1 above, *this* paper deals with our work towards appropriately discretizing euclidean n-space well for our purposes of simulating physical phenomena in a mesh computer based on such a discretization. In Section 1 above, we also indicated that the accuracy of this approach depends on such discretization of euclidean n-space, as well as suitable algorithms, in particular their timing, that could be run on such a mesh computer to simulate physical phenomena. It should be noted that such suitable algorithms are an important, integral part of this approach.

In [She91, CRS91c] we present an algorithm for simulating uniform radiation from a point source in bounded mesh computers based on lattices in general, and more "natural" algorithms in bounded mesh computers based on A_n and $\mathbf{Z}^n$, respectively. In [She91, CRS91b] we discuss mesh computers based on lattices from another class of lattices, the *well-represented* lattices. There, we also present an algorithm template for the simulation of constant speed particle motion in a mesh computer based on a well-represented lattice, and then an instance of this template for the simulation of constant speed particle translation in such mesh computers. In [She91] we present an algorithm, also an instance of the template from [She91, CRS91c], for the simulation of constant angular speed particle revolution. It turns out that the class of well-represented lattices intersects with the class of *root* lattices (see [CN88] for a discussion on root lattices). Several characterizations of root lattices are given in [RS91]. Interestingly, it also turns out that a direct consequence of one of these characterizations of root lattices is that the "natural" algorithm, mentioned just above, for uniform radiation from a point source can be generalized to work on a bounded mesh computer based on any root lattice.

References

[AG89] George S. Almasi and Allan Gottlieb. *Highly Parallel Computing*. The Benjamin/Cummings Publishing Company, Inc., 1989.

[AS84] N. Ahuja and S. Swamy. Multiprocessor pyramid architecture for bottom-up image analysis. *IEEE Trans. on PAMI*, PAMI-6:463–474, 1984.

[CN88] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer Verlag, 1988.

[CRS90a] J. Case, D. S. Rajan, and A. M. Shende. Optimally representing euclidean space discretely for analogically simulating physical phenomena. In K. V. Nori, editor, *Foundations of Software Technology & Theoretical Computer Science*, volume 472 of *Lecture Notes in Computer Science*, pages 190–203. Springer Verlag, Berlin, 1990. See [CRS91a].

[CRS90b] John Case, Dayanand S. Rajan, and Anil M. Shende. Parallel processor computer, Dec 1990. U.S. Patent Application submitted by SUNY, Research Foundation.

[CRS91a] John Case, Dayanand S. Rajan, and Anil M. Shende. Lattice computers for optimally representing euclidean space. Technical Report 91-15, University of Delaware, June 1991. Expands on and corrects slightly [CRS90a].

[CRS91b] John Case, Dayanand S. Rajan, and Anil M. Shende. Simulating uniform motion in lattice computers I: Constant speed particle translation. Technical Report 91-17, University of Delaware, June 1991.

[CRS91c] John Case, Dayanand S. Rajan, and Anil M. Shende. Spherical wave front generation in lattice computers. Technical Report 91-16, University of Delaware, June 1991.

[DL81] M. J. B. Duff and S. Levialdi, editors. *Languages and Architectures for Image Processing.* Academic Press, 1981.

[Dye] C. R. Dyer. A quadtree machine for parallel image processing. Technical Report KSL 51, Univ. of Illinois at Chicago Circle.

[Fey82] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7), 1982.

[FHP86] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice-gas automata for the Navier Stokes equation. *Physical Review Letters*, 56(14):1505–1508, April 1986.

[FT82] Edward Fredkin and Tommaso Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3/4), 1982.

[GL87] Peter M. Gruber and Gerrit Lekkerkerker. *Geometry of Numbers.* North-Holland Mathematical Library, 1987.

[Has87] Brosl Hasslacher. Discrete fluids. *Los Alamos Science*, (15):175–217, 1987. Special Issue.

[Her64] I. N. Herstein. *Topics in Algebra.* Blaisdell Publishing Co., 1964.

[Hil55] E. L. Hill. Relativistic theory of discrete momentum space and discrete space–time. *Physical Review*, 100(6), December 1955.

[Lev85] S. Levialdi, editor. *Integrated Technology for Parallel Image Processing.* Academic Press, Inc., 1985.

[Mac86] Thinking Machines. Introduction to data level parallelism. Technical Report 86.14, Thinking Machines, April 1986.

[Mar84] Norman Margolus. Physics–like models of computation. *Physica 10D*, pages 81–95, 1984.

[Min82] Marvin Minsky. Cellular vacuum. *International Journal of Theoretical Physics*, 21(6/7), 1982.

[PD84] Kendall Preston and M. J. B. Duff. *Modern Cellular Automata: Theory and Applications.* Plenum Publishers, 1984.

[PF+85] John Poulton, Henry Fuchs, et al. PIXEL–PLANES: Building a VLSI–based graphic system. In *Chapel Hill Conference on VLSI*, 1985.

[PU82] Kendall Preston and Leonard Uhr, editors. *Multicomputers and Image Processing: Algorithms and Programs.* Academic Press, 1982.

[RS91] Dayanand S. Rajan and Anil M. Shende. Efficiently generated lattices and root lattices. Technical Report 92-02, University of Delaware, August 1991. Journal article under review.

[She91] Anil M. Shende. *Digital Analog Simulation of Uniform Motion in Representations of Physical n-Space by Lattice-Work MIMD Computer Architectures.* PhD thesis, SUNY at Buffalo, Buffalo, N. Y., 1991. TR 91-14, Department of Computer Science.

[Sny47] Hartland S. Snyder. Quantized space–time. *Physical Review*, 71(1):38–41, 1947.

[Svo86] Karl Svozil. Are quantum fields cellular automata? *Physics Letters A*, 119(4), December 1986.

[SW86] James B. Salem and S. Wolfram. Thermodynamics and hydrodynamics with cellular automata. In S. Wolfram, editor, *Theory and Applications of Cellular Automata.* World Scientific, 1986.

[TM87] Tommaso Toffoli and Norman Margolus. *Cellular Automata Machines.* MIT Press, 1987.

[Tof77a] Tommaso Toffoli. Cellular automata machines. Technical Report 208, Comp. Comm. Sci. Dept., University of Michigan, 1977.

[Tof77b] Tommaso Toffoli. Computation and construction universality of reversible cellular automata. *Journal of Computer and System Sciences*, 15:213–231, 1977.

[Tof84] Tommaso Toffoli. CAM: A high–performance cellular–automaton machine. *Physica 10D*, pages 195–204, 1984.

[Uhr72] L. Uhr. Layered 'recognition cone' networks that preprocess, classify, and describe. *IEEE Trans. on Computers*, C-21:758–768, 1972.

[Ung58] S. H. Unger. A computer oriented towards spatial problems. In *Proceedings of the IRE*, volume 46, pages 1744–1750, 1958.

[Vic84] Gérard Y. Vichniac. Simulating physics with cellular automata. *Physica 10D*, pages 96–116, 1984.

[Wol83] Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601–644, July 1983.

[Zus69] K. Zuse. *Rechnender Raum*. Vieweg, Braunshweig, 1969. Translated as Calculating Space, Tech. Transl. AZT–70–164–GEMIT, MIT Project MAC, 1970.