1991

A Spatial Index for Convex Simplicial Complexes in d Dimensions

George Vaněček

Vincenzo Ferrucci

Report Number:
91-001

Vaněček, George and Ferrucci, Vincenzo, "A Spatial Index for Convex Simplicial Complexes in d Dimensions" (1991). *Department of Computer Science Technical Reports*. Paper 850.
<https://docs.lib.purdue.edu/cstech/850>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries.
Please contact epubs@purdue.edu for additional information.

**A SPATIAL INDEX FOR CONVEX SIMPLICIAL
COMPLEXES IN d DIMENSIONS**

George Vanacek, Jr.
Vincenzo Ferrucci

CSD-TR-91-001
January 1991

A SPATIAL INDEX FOR CONVEX SIMPLICIAL COMPLEXES IN d DIMENSIONS¹

George Vaneček, Jr.²
and
Vincenzo Ferrucci³

Computer Sciences Department
Purdue University
Technical Report CSD-TR-91-001
CAPO Report CER-91-01
January, 1991

¹ This research was supported, through the Leonardo Fibonacci Institute, by the Istituto Trentino di Cultura, Trento, Italy.

² Additional support was provided by NSF grant CCR-86-19817 to Purdue University.

³ Additional support was provided by PF "Edilizia" and "Robotica" of the Italian National Research Council.

A Spatial Index for Convex Simplicial Complexes in d Dimensions¹

George Vaněček, Jr.²
Dep. of Computer Science
Purdue University
West Lafayette, IN 47907

Vincenzo Ferrucci³
Dip. di Informatica e Sistemistica
Università "La Sapienza"
Via Buonarroti 12, 00184 Roma

January 2, 1991

Abstract

A data structure for representing convex simplicial complexes in d -dimensional space is presented with which a k -face of the complex containing a query point can be determined in time $O(\log^d nd)$. This is an improvement over a sequential search approach requiring $O(nd^3)$ time. We show that a structure of $O(d!n^d)$ storage can be easily constructed and present a compression technique which is conjectured to produce a structure of $O(nd2^d)$ storage.

¹This research was supported, through the Leonardo Fibonacci Institute, by the Istituto Trentino di Cultura, Trento Italy.

²Additional support was provided by NSF grant CCR-86-19817 to Purdue University.

³Additional support was provided by PF "Edilizia" and "Robotica" of the Italian National Research Council.

1 Introduction

Solid modeling was traditionally devoted to the representation and manipulation of 3D objects. Now however, solid modeling is moving to consider multidimensionality. The applications of multidimensional solid modeling are found in robotics, simulation, and scientific visualization.

Fundamental research in multidimensional solid modeling has been carried out by several research groups. For example, Putnam and Subrahmanyam [PS86] describe boolean operations between multidimensional objects. Rossignac and O'Connor [ROC89] are developing a representational tool with characteristics of great generality. Their scheme, SGC (Selective Geometric Complex), is dimension independent and based on the use of cell complexes, where a cell is a general manifold. Key to the method presented in this paper is the research conducted by the CAD group at the Dipartimento di Informatica of the University of Rome "La Sapienza". This project uses tools from algebraic topology as a conceptual reference frame which guarantee mathematically well founded developments. The employed representation scheme, the winged representation, is based on simplicial complexes [PC90] and allows the development of efficient algorithms for many basic operations, as for example, extrusion, boundary evaluation, and computation of integrals [FP90, Ber90].

When a representation scheme such as the winged representation is used, for example, to check the interference between moving objects, a more efficient representation is needed. Checking the interference is characteristic of a class of problems for which the representation is static and undergoes numerous spatial queries to the various k -faces of a complex. In this context it is useful to have a fast algorithm for classifying geometrical entities (points, lines, simplices) against a given simplicial complex.

This paper presents a data structure, called the *spatial index*, to represent a convex simplicial complex in $\mathbf{E}^d$ with which numerous spatial queries can be performed efficiently.

The spatial index is an extension of Lee and Preparata's chain method for locating points in a planar subdivision [LP77]. (An overview of the chain method can be found in [PS85, pp48–56]). They show that point-location in an N -vertex planar subdivision can be affected in $O(\log^2 N)$ time using $O(N)$ storage, given $O(N \log N)$ preprocessing time.

The spatial index for a d -dimensional simplicial complex can be constructed with which a k -face of the complex containing a query point can be determined in

time $O(\log^d nd)$. Without the use of the spatial index, a simple sequential search would require $O(nd^3)$ time. We show that a spatial index of $O(d!n^d)$ storage can be easily constructed and present a compression technique which is conjectured to be $O(nd2^d)$ storage.

The remainder of this section offers the necessary background on the winged representation and notation and a brief overview of our method. Section 2 start with an algorithm that orders the simplices and finds the separation sheets. The construction of the spatial index is given in the two subsequent sections. In Section 3, a full spatial index is constructed and the point/complex classification is given. A full spatial index is easy to construct, but replicates much of the complex and thus results in a large data structure. We present the full spatial index as a way to simplify the presentation of the compressed spatial index. In Section 4 the compressed spatial index is constructed and the point/complex classification is discussed. The data structure presented here is discussed and summarized in Section 5. We present a simple example of a full spatial index in the appendix.

1.1 Definitions

A k -simplex σ , for $\sigma \subset \mathbb{E}^d$, is a convex combination of $k + 1$ affinely independent points called vertices. The set of vertices $\{v_0, \dots, v_d\}$ is called a 0 -skeleton of the simplex. When the k -simplex is generated from any subset of $k + 1$ vertices of a d -simplex, for $0 \leq k \leq d$, it is called an k -face. A $(d - 1)$ -face of a d -simplex is simply referred to as a face. Two d -simplices are k -adjacent when they share a common k -face.

A *simplicial complex* $\Sigma^d = \{\sigma_1, \dots, \sigma_n, \dots\}$ of dimensions d , for $d > 0$, is the set of k -simplices, for all $0 \leq k \leq d$, having the conditions: (a) if $\sigma \in \Sigma^d$, then any face of σ belongs to Σ^d ; (b) if $\sigma, \tau \in \Sigma^d$, then either $\sigma \cap \tau = \emptyset$, or $\sigma \cap \tau$ is a k -face of σ or τ . A simplicial complex is *regular* when each k -simplex is a k -face of a d -simplex.

An ordering of the 0 -skeleton of a simplex implies an orientation of the simplex. Two $(d - 1)$ -adjacent d -simplices are coherently oriented when their common faces have opposite orientation. The simplicial complex is *coherently oriented* when each $(d - 1)$ -adjacent pair of d -simplices is coherently oriented. From now on, any simplicial complex is assumed to be regular and coherently oriented.

We denote the s -skeleton of Σ^d as $K^s(\Sigma^d)$, for $0 \leq s \leq d$, defined as the set of the s -faces of Σ^d .

The *boundary complex* of Σ^d , $\partial\Sigma^d$, is a simplicial complex with $(d-1)$ -simplices $\{\tau_1, \dots, \tau_m\}$ which are the faces of exactly one d -simplex in Σ^d , along with all the k -faces of τ_i , for $0 \leq k < d$, and $1 \leq i \leq m$. Thus $\partial\Sigma^d \subset \Sigma^d$. Note that this definition differs from [FP90].

A *hyperplane* in $\mathbf{E}^d$ is the locus of all points $p = (x_1, \dots, x_d)$ satisfying an equation

$$a_0 + \sum_{i=1}^d a_i \cdot x_i = 0, \quad (1)$$

where $a_0, \dots, a_d$ are the coefficients in $\mathbf{R}$.

Given that Σ^d has a k -simplex of k at most $d-1$, and that Σ^d is monotone in a given direction (*monotonicity* means that all lines in direction of the projection intersects the complex in at most one k -simplex, for any $k < d$), the *projection* of Σ^d onto the coordinate hyperplane $x_i = 0$, $\Pi_i(\Sigma^d)$, is defined as the set

$$\Pi_i(\Sigma^d) = \{\Pi_i(\sigma) \mid \sigma \in \Sigma^d\}$$

in which $\Pi_i(\sigma)$ is the convex combination of vertices $\langle v'_0, \dots, v'_s \rangle$ with

$$v'_j = (x_0, \dots, x_{i-1}, x_{i+1}, \dots, x_s)$$

obtained from the 0-skeleton of the s -simplex, σ .

1.2 Representation

We note that the d -skeleton, $K^d(\Sigma^d)$, is a complete representation of Σ^d , as all the remaining s -skeletons of Σ^d for $0 \leq s < d$, can be obtained from the d -skeleton. Given this, we base our spatial index on the winged representation $\mathcal{W}$ of Paoluzzi and Cattani [PC90], which is based primarily on the d -skeleton. The representation is used by a multidimensional modeling system called *Simplex*_X [Fer90]. A formal framework and further details on the representation can be found in [PC90].

Definition 1 Let $\mathcal{W}$ be the winged representation of a simplicial complex Σ^d , consisting of a finite set of vertices, $K^0(\Sigma^d)$, and a set of d -simplices, $K^d(\Sigma^d)$, where $\sigma_i \in K^d(\Sigma^d)$ has the representation

$$(S(\sigma_i), V(\sigma_i), A(\sigma_i)) = (\pm 1, \langle v_{i0}, v_{i1}, \dots, v_{id} \rangle, \langle a_{i0}, a_{i1}, \dots, a_{id} \rangle)$$

consisting of the orientation sign, $S(\sigma_i)$, given as $+1$ or -1 , an ordered list of $d+1$ vertices of the simplex, $V(\sigma_i)$, and a list of $d+1$ adjacent d -simplices, $A(\sigma_i)$, with $a_{ij} \in K^d(\Sigma^d) \cup \{\emptyset\}$. If $a_{ij} = \emptyset$, then there is no adjacent simplex to σ_i along the corresponding face.

The vertices of the j th face of σ_i are to be obtained from $V(\sigma_i)$ by removing the j th vertex, and as such we use an additional subscript on a simplex to denote the face, i.e.,

$$V(\sigma_{ij}) = \langle v_0, \dots, v_{j-1}, v_{j+1}, \dots, v_d \rangle.$$

The normal vector, $\mathbf{n}_{ij}$, of the oriented support hyperplane of the face is obtained as the normalized and properly oriented form of the vector $(a_1, \dots, a_d)$ which consists of the coefficients of the hyperplane given by Equation (1). The coefficients are obtained by solving the minors of the $(d+1) \times (d+1)$ determinant

$$\begin{vmatrix} a_1 & \cdots & a_d & 1 \\ x_{01} & \cdots & x_{0d} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_{i-1,1} & \cdots & x_{i-1,d} & 1 \\ x_{i+1,1} & \cdots & x_{i+1,d} & 1 \\ \vdots & \ddots & \vdots & \vdots \\ x_{d1} & \cdots & x_{dd} & 1 \end{vmatrix} = 0 \quad (2)$$

where $v_j = (x_{j0}, \dots, x_{jd})$, $0 \leq j \leq d$.

It is instructive at this point to look at the space requirements for $\mathcal{W}$ and at the time requirement for performing a point/complex classification on $\mathcal{W}$ without any preprocessing.

A set of n d -simplices forms a simplicial complex called a *chain* when $n-2$ of its d -simplices are $(d-1)$ -adjacent to exactly two other d -simplices and two of its d -simplices, one at each end of the chain, are $(d-1)$ -adjacent to only one other simplex. The maximum storage for a connected d -complex, Σ^d , is needed when Σ^d is a chain, that is, n d -simplices, and $n+d$ vertices. Representation $\mathcal{W}$ thus requires at most

$$(n+d)d + 2n(d+1) = O(dn + d^2)$$

space.

Without preprocessing, the point/complex classification is relatively straightforward. Since it is not known which simplex contains the point and the intermediate k -faces are not represented explicitly, it is necessary to check each d -simplex for the point. Once the d -simplex containing the point is found, the k -face containing the query point p can be determined by solving a system of $d+1$ equations and unknowns, and selecting all the vertices corresponding to

non-zero solutions. More precisely, a point p in or on a d -simplex with vertices $\langle v_0, \dots, v_d \rangle$ can be defined in terms of its convex coordinates, (λ_i) , leading to the following $d + 1$ equations:

$$\begin{aligned} \sum_{0 \leq i \leq d} v_i \cdot \lambda_i &= p \\ \sum_{0 \leq i \leq d} \lambda_i &= 1 \end{aligned}$$

So for a given d -simplex, the system of equations can be represented in a $(d + 1) \times (d + 2)$ matrix and solved in $O(d^3)$ time. The following theorem summarizes the results.

Theorem 1 *Given the simplicial complex $\mathcal{W}$ requiring $O(dn + d^2)$ storage, with no preprocessing, the point/complex classification takes $O(nd^3)$ time and $O(d^2)$ additional space,*

1.3 Overview of the Spatial Index

Without loss of generality, we require the input simplicial complex to have the property that the projection of any k -face, for $1 \leq k \leq d - 1$, onto all the coordinate hyperplanes in $\mathbb{E}^d$ is a k -simplex. That is, the dimension of the k -face is preserved by the projection. Complexes for which this property does not hold can be slightly perturbed by a small rotation to establish the property. We also require that the complex be convex. The cavities of a nonconvex complexes can initially be triangulated to establish convexity, although this may not be practical in higher dimensional space.

The construction of the spatial index for Σ^d , relies on the ability to partition the set of d -simplices of Σ^d into two roughly equal size sets. The sets are separated by a $(d - 1)$ -complex that is monotone, by convention, with respect to one of the d coordinate hyperplanes. We refer to the set of $(d - 1)$ -faces that partition the d -complex as the *separation sheet*, Ψ . Because of the monotonicity, a point in $\mathbb{E}^d$ can be determined to lie above, on, or below the separation sheet, thereby reducing the search of the set of the d -simplices by half. A recursive partitioning of the n d -simplices of the complex yields a set of separation sheets $(\Psi_0, \dots, \Psi_n)$. This partitioning presupposes an ordering of the d -simplices, $\{\sigma_{\pi_1}, \dots, \sigma_{\pi_n}\}$, so that Ψ_i partitions the set of simplices into the set of simplices $\{\sigma_{\pi_1}, \dots, \sigma_{\pi_i}\}$ below the sheet, and the set of simplices $\{\sigma_{\pi_{i+1}}, \dots, \sigma_{\pi_n}\}$ above the sheet.

Figure 1: The structure of an internal node r_m of the spatial index.

This ordering allows the d -simplices and the separation sheets to be organized into a binary search tree with $n + 1$ internal nodes, one for each separation sheet, and $n + 2$ leaves, one for each simplex and two for the regions above and below the complex. Note that the projections of the separation sheets are the $(d - 1)$ -skeletons of the $(d - 1)$ -complexes. Since these have to be searched during the traversal of the binary tree in the d dimension, the sheets are recursively constructed into binary search trees in the $d - 1$ dimension and form the middle subtrees in the d -dimensional binary search tree. Figure 1 shows the structure of an internal node r_m , for $1 \leq m \leq n$.

With the spatial index, a point can be efficiently classified by using the separation sheets to direct the binary search within a given dimension. At each internal node, the search proceeds down the left or the right subtrees depending on whether the point is above or below the separation sheet. The comparison of the point and the separation sheet is handled recursively by projecting both the point and the sheet, and finding the face of the sheet that contains the projected point. Knowing the face allows the comparison of the point against the support hyperplane of the face, as shown in Figure 2 for a $2D$ example.

2 Finding the Separation Sheets in E^d

For the construction of the spatial index, the separation sheets are formed conceptually by sweeping a rubber sheet through the complex. The sweep proceeds from one extreme of the complex to the other by an iterative process of including the simplices of the complex. Initially, a separation sheet is computed that separates the complex from the region below the complex. At each step, the next separation sheet is obtained from the previous separation sheet by removing the

Figure 2: Comparing point p against the oriented support line (shown dashed) of edge e .

lower faces and including the upper faces of the next simplex.

This conceptual construction of the separation sheets requires $O(d^2n^2)$ space in the worst case—consider the chain example of Section 1.2. However since it is not necessary to know all the separation sheets at once, the sweep technique can be replaced by one in which a separation sheet is computed only as needed, that way, only $O(dn)$ additional space is used.

Both methods depend on the lower and the upper envelopes of a complex. An envelope consists of the faces on the boundary of a complex for which the normals of the support hyperplanes are oriented in the proper direction. Thus, the normals of the oriented support hyperplanes of the $(d - 1)$ -simplices of the lower envelope point in the direction $\mathbf{n}$, and, $-\mathbf{n}$ in the case of the upper envelope.

Definition 2 Given Σ^d and $\mathbf{n}$, the projection direction, the lower envelope of Σ^d is a set of faces $\{\sigma_{ij} \in K^{d-1}(\partial\Sigma) \mid \mathbf{n}_{ij} \cdot \mathbf{n} > 0\}$, and the upper envelope of Σ^d is a set of faces $\{\sigma_{ij} \in K^{d-1}(\partial\Sigma^d) \mid \mathbf{n}_{ij} \cdot -\mathbf{n} > 0\}$.

For the sheets to be monotone, the simplices of the complex must be ordered in the opposite direction of projection. Figure 3 show a 2-complex with eight triangles, sixteen edges and nine vertices. The label in the triangles indicates the order number. Based on the ordering, the nine separation sheets are:

$$\begin{aligned}
 \Psi_0 &= \{a, b, c, d\} & \Psi_1 &= \{e, f, b, c, d\} \\
 \Psi_2 &= \{e, f, g, h, c, d\} & \Psi_3 &= \{e, i, h, c, d\} \\
 \Psi_4 &= \{e, i, j, d\} & \Psi_5 &= \{e, k, l, j, d\} \\
 \Psi_6 &= \{e, k, l, m, n, d\} & \Psi_7 &= \{e, k, o, n, d\} \\
 \Psi_8 &= \{e, k, o, p\}
 \end{aligned}$$

Figure 3: Example of a two-dimensional simplicial complex with eight triangles, sixteen edges, and nine vertices labeled A through I.

In the remainder of this section the algorithm for ordering the d -simplices and both methods for obtaining the separation sheets are presented.

2.1 Ordering the d -simplices

Based on the normal vectors of the faces that separate adjacent simplices, we can construct a partial ordering of the complex such that no simplex σ can precede another simplex τ , if any portion of σ obscures τ in the projection direction. From the partial ordering the desired total ordering of the simplices can be obtained by a topological sort. Clearly such an ordering exists for $d < 3$. For higher dimensions, a presence of cycles may exist, and either a different projection direction must be chosen or the cycle must be broken by triangulating one or more of the simplices in the cycle. Here, we assume that no cycles exist.

Algorithm SSC: (Sort Simplicial Complex) Let Σ^d be the simplicial complex. Then this algorithm orders the d -simplices of Σ^d in the direction $-\mathbf{n}$ producing permutation π .

1. Define the projection direction $\mathbf{n} = -\mathbf{e}_d$, where $\{\mathbf{e}_1 \dots \mathbf{e}_d\}$ is the standard basis for $\mathbf{E}^d$ (i.e., $\mathbf{n} = (0, \dots, 0, -1)$).
2. Construct a directed graph $G = (V, E)$ with vertices, V , labeled by the

d -simplices of Σ^d , and with a directed edge, $(\sigma_i, \tau) \in E$, if σ_i is $(d-1)$ -adjacent to τ along a face, σ_{ij} , and its normal $\mathbf{n}_{ij}$ is oriented in the same direction as $\mathbf{n}$.

3. Given that G is a directed acyclic graph, an ordered set of the d -simplices of Σ^d given by the permutation $(\pi_1, \dots, \pi_n)$ is obtained by a topological sort [AHU83] of G .

In Step 2 of Algorithm SSC, the normal of the support hyperplane of a $(d-1)$ -face is computed, and the direction of its normal is checked against the projection direction. The normal of the face is in the same direction as the projection direction, when $\mathbf{n} \cdot \mathbf{n}_{ij} > 0$. This computation requires the solution to the determinant of Eq. (2) to obtain $\mathbf{n}_{ij}$. However, since $\mathbf{n}$ has only one nonzero element, it suffices to solve only two minors of the determinant, which takes $O(d^3)$. Now since the graph has n nodes and less than $(d+1)n/2$ arcs, the topological sort takes $O(n + dn)$ time, which gives the algorithm $O(d^4n)$ time and space.

2.2 Computing all the Separation Sheets

Taking the lower envelope and flipping the orientation of its faces reverses the normals and gives us the separation sheet Ψ_0 ; for Ψ represented by $\mathcal{W}$ it suffices to complement the sign of each simplex. Starting with Ψ_0 , a sweep through the complex in the direction $-\mathbf{n}$ yields the remaining separation sheets, as follows:

Algorithm FSS: (Full Separation Sheets) Let $(\pi_1, \dots, \pi_n)$ be the permutation for the d -simplices of Σ^d as computed by Algorithm SSC. This algorithm returns $(\Psi_0, \dots, \Psi_n)$.

1. Obtain the lower envelope and positively reorient it to form Ψ_0 of Σ^d .
2. Obtain the intermediate separation sheets as

$$\Psi_i = (\Psi_{i-1} \cup K^{d-1}(\sigma_{\pi_i})) - (\Psi_{i-1} \cap K^{d-1}(\sigma_{\pi_i})) \quad (3)$$

for $1 \leq i \leq n$.

Lets consider the time complexity of Algorithm FSS. In Step 1, Ψ_0 is the subset of $K^{d-1}(\partial\Sigma^d)$ with faces that have the normals of their support hyperplanes in the direction of projection. The boundary faces can be identified in $O(dn)$ steps, and the orientation of each can be checked in $O(d^3)$. For a chain with n

Figure 4: A 2D example of a *chain*.

d -simplices, the number of faces in the boundary is $(d+1)n - 2(n-1)$, since the first and the last simplex in the chain lose a single face and all other simplices lose two faces to the adjacency with other touching simplices (refer to Figure 4). Therefore, the number of faces in a separation sheet Ψ is bounded by

$$1 \leq |\Psi| \leq dn - n + 1.$$

That is, the lower envelope can be a single face, or as large as $dn - n + 1$ if the d -complex is a chain for which all but one face is visible in the projection direction. Thus Step 1 needs $O(d^4n)$ time and $O(dn)$ space.

Step 2 iteratively creates the intermediate separation sheets. The separation sheet Ψ_{i+1} is created by expanding the upper envelope, Ψ_i , of $(\sigma_{\pi_1}, \dots, \sigma_{\pi_i})$, with the upper envelope of the next simplex, $\sigma_{\pi_{i+1}}$. Since the number of $(d-1)$ -simplices in a subsequent separation sheet can vary by at most d , and the set of simplices can be updated in a single pass, each iteration of Step 2 can be performed in time $O(dn)$.

Theorem 2 *Algorithm FSS requires $O(d^4n + d^2n^2)$ time and $O(dn^2)$ space to obtain all $n + 1$ separation sheets.*

2.3 Obtaining a single Separation Sheet

The sheet Ψ_i is easily obtained as the union of the upper envelope of the lower complex $(\sigma_{\pi_1}, \dots, \sigma_{\pi_i})$, and the properly reoriented lower envelope of the upper complex $(\sigma_{\pi_{i+1}}, \dots, \sigma_{\pi_n})$.

Lemma 1 *Given Σ^d with the d -simplices ordered along the projection direction $\mathbf{n}$, then*

$$\Psi_i = -g(\Sigma_U, \mathbf{n}) \cup g(\Sigma_L, -\mathbf{n}),$$

Figure 5: The upper and the lower envelopes of the simplicial complex of Figure 1 that are needed for the separation sheet Ψ_4 .

for $0 \leq i \leq n$, where

$$g(\Sigma, \mathbf{v}) = \{\sigma_{ij} \in K^{d-1}(\partial\Sigma^d) \mid \mathbf{n}_{ij} \cdot \mathbf{v} > 0\},$$

$$\Sigma_L = (\sigma_{\pi_1}, \dots, \sigma_{\pi_i}), \text{ and } \Sigma_U = (\sigma_{\pi_{i+1}}, \dots, \sigma_{\pi_n}).$$

Proof: By definition, Ψ_i is the subset of $(d-1)$ -simplices of $\partial\Sigma^d$, such that each d -simplex in Σ_L is below Ψ_i and each simplex Σ_U is above Ψ_i . This is guaranteed by the construction. Notice that it is necessary to reorient $g(\Sigma_U, -\mathbf{n})$ for Ψ_i to be oriented coherently. $\square$

For example, Figure 5 shows the upper and the lower envelopes of the two sub-complexes needed for the separation sheet Ψ_4 of the 2-complex in Figure 3.

The algorithm for computing a desired separation sheet follows from the lemma directly.

Algorithm SS: (Separation Sheet) Given Σ^d , $(\pi_1, \dots, \pi_n)$, and $0 \leq i \leq n$, return Ψ_i .

1. Let $\Sigma_L = (\sigma_{\pi_1}, \dots, \sigma_{\pi_i})$, and $\Sigma_U = (\sigma_{\pi_{i+1}}, \dots, \sigma_{\pi_n})$.
2. Compute the lower envelope $\Psi_L = g(\Sigma_U, \mathbf{n})$ and the upper envelope $\Psi_U = g(\Sigma_L, -\mathbf{n})$, where $g(\Sigma, \mathbf{v})$ chooses all the faces on the boundary of Σ^d with normals of the support hyperplanes oriented in direction $\mathbf{v}$.
3. Return $\Psi_i = -\Psi_L \cup \Psi_U$.

Theorem 3 *Algorithm SS takes $O(d^4n + dn \log dn)$ time and $O(dn)$ space.*

Proof: Step 1 takes $O(1)$. Step 2 looks at every boundary face once and computes its normal, thus taking $O(d^4n)$. Since there are $O(dn)$ faces in Ψ_i , the union in Step 3 can be performed in $O(dn \log dn)$ time. $\square$

3 The Full Spatial Index

We now construct the spatial index given the simplicial complex as a winged representation of Definition 1. The algorithm follows:

Algorithm BFI: (Build Full Index) Let Σ^d , for $d \geq 0$, be a convex simplicial complex given the assumptions of Section 1.3. The algorithm returns the full spatial index for Σ^d .

1. If $d = 0$ then return the single 0-simplex of Σ^0 and skip the remaining steps.
2. With Algorithm SSC, order the d -simplices of Σ^d and obtain the permutation $(\pi_1, \dots, \pi_n)$.
3. Create a balanced binary tree T with the fringe (refer to Figure 6)

out, $\sigma_{\pi_1}, \dots, \sigma_{\pi_n}$, out.

4. If d is not the highest dimension of the original complex, then associate with each simplex σ_i the support hyperplane of Eq. (1) in $\mathbf{E}^{d+1}$, which is positively oriented so that $a_d > 0$.
5. For each internal node r_m (refer to Figure 1), compute Ψ_m by either Algorithm SS or FSS, and associate with node r_m the spatial index of $\Pi_d(\Psi_m)$ computed recursively from Step 1.
6. Return the spatial index T .

The algorithm works recursively on the dimensions of the complex. The terminating condition of Step 1 occurs when the given complex is zero dimensional. In that case, the result is a trivial tree consisting of the single 0-simplex of the complex.

For higher dimensional complexes, a nontrivial structure results. The dimension d of the complex imposes the projection direction $-\mathbf{e}_d$, and in Step 2, the simplices are ordered linearly along $\mathbf{e}_d$ by Algorithm SSC.

The ordered simplices along with the two bounding **out** regions yield a binary search tree in Step 3. The binary tree has $n + 1$ internal nodes with the fringe of size $n + 2$. The binary tree is easily constructed by recursively subdividing the ordered list

out, $\sigma_{\pi_1}, \dots, \sigma_{\pi_n}$, out,

Figure 6: The spatial index for the example given in Figure 3. The middle subtrees, $\Pi_2(\Psi_i)$, are indicated by triangles and are not shown.

so that the inorder traversal of the binary tree yields

$$\text{out}, \Psi_0, \sigma_{\pi_1}, \Psi_1, \dots, \sigma_{\pi_n}, \Psi_n, \text{out}.$$

An example showing the search tree of the complex of Figure 3 is found in Figure 6.

In Step 4, the support hyperplanes for the simplices in the fringed are computed as described in Section 2 and stored. Note that the hyperplanes are oriented so that their normal is pointing in the opposite direction as the projection direction, that is, in the direction $\mathbf{e}_d$. This step is not necessary for the highest dimension of the original complex.

In Step 5, for each internal node r_i the separation sheet Ψ_i is computed. Its projection along $\mathbf{e}_{d-1}$ defines a subcomplex that is used to recursively construct the spatial index associated with node r_i .

Consider the size of the constructed full spatial index. Since each separation sheet for every dimension is represented in its entirety, and a sheet can have as many as $nd - n + 1$ faces, the number of nodes must be

$$(n+1) \prod_{k=1}^d [n(d-k) + 1] = O(d!n^d).$$

Before proceeding with the presentation of the compressed spatial index we first show that, even in this full spatial index, a point can be classified in time $O(\log^d dn)$.

3.1 Point/Complex Classification

The full spatial index allows a point to be classified in an efficient manner. The classification determines in which k -simplex, for $0 \leq k \leq d$, a given point lies. Conceptually, the classification traverses a single path from the root to a leaf of the binary tree for a given dimension and recurses down through lower dimensions. When a query point projects only through $(d - 1)$ -faces of a complex, the classification is trivial. At every step for some dimension k , the recursive step returns a $(k - 1)$ -face. The comparison always yields an above or below result, and the search proceeds down to a k -simplex in a fringe. When however, the query point projects onto some k -face, for $k < d - 1$, at dimension $k + 2$, the classification of the point must be performed against a $(k + 1)$ -face, not the k -face. As such, the $k + 1$ -dimensional result must return both the k -face and one of the adjacent $(k + 1)$ -face.

Algorithm PC1: (Point/Complex Classification) Let r_m be the root node of the full spatial index for Σ^d and p , the point to be classified. Return **out** if the point is outside the complex, or return the k -face of Σ^d that contains p , for a minimum k .

1. If node r_m represents either **out** or a k -face, then return $[r_m, \emptyset]$.
2. Let Ψ_m be the separation sheet of node r_m . Recursively find the faces $[f, f']$ where $f \in \Psi_m$ for $\Pi_d(p) \in \Pi_d(f)$, and if p projects onto a k -face, f' is adjacent to f for $k > d - 1$.
3. Compare p against the hyperplane of f and proceed depending on whether p is above, on or below as follows:

Above Classify p starting from Step 1 with the *above* child node of r_m .

On Return either $[\sigma_m, f']$ if $m \geq 1$ or $[\sigma_1, f']$ if $m = 0$.

Below Classify p starting from Step 1 with the *below* child node of r_m .

The termination condition is reached when the complex is zero dimensional or when the binary search in a given dimension falls into one of the two **out** regions.

When the search is entered at some internal node r_m at Step 2, a decision has to be made as to whether to proceed the binary search down the *above*, or the *below* subtree. This depends on the relative position of the query point in

relation to the separation sheet Ψ_m . For this, a face f is found recursively in the separation sheet onto which the query point projects.

In Step 3, the binary search down the *above* or *below* subtrees, or returns if p is found to lie on f .

Theorem 4 *A k -face of a complex containing a query point can be located by Algorithm PC1 in time $O(\log^d dn)$.*

Proof: The search traces down a binary search tree for each dimension.

Since each binary search tree contains $O(dn)$ nodes, a given dimension requires $O(\log dn)$. Therefore for d dimensions, the search is $O(\log^d dn)$. $\square$

4 Compressed Spatial Index

In the previous section we constructed a full spatial index in which much of the information maintained in the separation sheets is redundant. In this section, we construct a compressed spatial index and present a modified point/complex classification algorithm that uses the compressed spatial index.

From Eq. (3), we see that Ψ_i and Ψ_{i+1} are almost identical except for the faces of one simplex. To reduce the storage needed for the sheets in a given dimension, we have to eliminate all redundancies in the sheets. In all the separation sheets for a given dimension, a face needs to appear only once. By removing redundant faces, the partial sheets that are stored become sparse the deeper they are in the tree. In the lower levels of the tree, the sheets may become completely empty. Conceptually, the complete sheets are implicit in the search path from the root down to a leaf.

Let the separation sheet Ψ_i have a representation inside a spatial index tree T as Ψ_i^T . Furthermore, let $(k_1, k_2, \dots, k_m)$, for $k_1 = \lfloor (n+1)/2 \rfloor$ and $k_m = i$ indicate the path from the root of T to the node containing Ψ_i . We let

$$\Psi_i^T = \Psi_i - (\Psi_{k_1}^T \cup \dots \cup \Psi_{k_m}^T). \quad (4)$$

It follows that $\Psi_{k_1}^T = \Psi_{\lfloor (n+1)/2 \rfloor}$ is the root-node separation-sheet, and that $\Psi_{k_m}^T$ may be an empty set. For example, the nine separation sheets for the tree of Figure 6 can be reduced to

$$\begin{array}{lll} \Psi_0^T & = & \{a\} \\ \Psi_3^T & = & \{\} \\ \Psi_6^T & = & \{k, l, m, n\} \end{array} \quad \begin{array}{lll} \Psi_1^T & = & \{b\} \\ \Psi_4^T & = & \{e, i, j, d\} \\ \Psi_7^T & = & \{o\} \end{array} \quad \begin{array}{lll} \Psi_2^T & = & \{f, g, h, c\} \\ \Psi_5^T & = & \{\} \\ \Psi_8^T & = & \{p\} \end{array}$$

The compressed spatial index can be constructed in two ways. The first way constructs a full spatial index using Algorithm BFI and then prunes the tree in a post processing step. The second way builds a compressed spatial index by modifying Algorithm BFI so that in Step 5, Ψ_m^T of Eq. (4) is used instead of Ψ_m , and so that Algorithms SSC and SS take the incomplete sheets into account. Although we are striving for the second way, the first way is easier to implement.

We observe that for a given k , the number of k -faces of a d -simplex is

$$\binom{d+1}{d-k}.$$

Thus the total number of k -faces of a d -simplex, is

$$\sum_{k=0}^d \binom{d+1}{k+1} = 2^{d+1} - 1.$$

If each k -face is represented by an ordered list of its $k+1$ vertices, then a single d -simplex needs

$$\sum_{k=0}^d (k+1) \binom{d+1}{k+1} + d(d+1) = (d+1)2^d + O(d^2),$$

where $d(d+1)$ is the additional storage for the $d+1$ vertices of the simplex. Now since in a fully compressed spatial index every k -face appears only once and the number of internal nodes is linearly proportional to the fringe, the size of the ternary tree must be at least $\Omega(nd2^d)$. This we conjecture is the size of the compressed spatial index.

4.1 Point/Complex Classification

The point/complex classification given a compressed spatial index proceeds identically to the way Algorithm PC1 searched a full spatial index except that some separation sheets need not be searched. Consider a binary tree of a given dimension with separation sheets Ψ_j and Ψ_k , for $j < k$ such that node r_k lies on the path from the root to node r_j , as shown in Figure 7. Given that $\Psi_j^T \subset \Psi_k$, there is at least one missing face in Ψ_j^T that appears in Ψ_k^T . If a query point projects through that face, the classification with Ψ_j^T would encounter a hole in the sheet, and therefore, the algorithm must guarantee that the point is not compared against Ψ_j^T , and that when visiting node r_j , it is known a priori which way to proceed. The Hole Theorem provides the necessary means.

Figure 7: Correctness proof for searching a compressed spatial index.

Theorem 5 (Hole) *Let $f \in \Psi_k$ and $f \in \Psi_j$, for $j < k$ such that node r_k lies on the path from the root to node r_j . If p lies below the support hyperplane of f , then either $p \in \sigma_i$ for $i < j$, or $p \notin \Sigma^d$.*

Proof: Since p is found to lie below $f \in \Psi_k$, then suppose that f is adjacent to $\sigma_{\pi_\alpha} \in \Sigma^d$ lying below Ψ_k as shown in Figure 7. It follows that p lies in or below σ_{π_α} . Now, since f is also in Ψ_j , then p must lie in a simplex equal to or lower than σ_{π_j} , and $\alpha \leq j$. Thus $p \in \sigma_i$ for $i < j$, or p is below the complex and therefore does not lie in any simplex. $\square$

From the Hole Theorem, the next corollary follows directly by symmetry.

Corollary 1 *Let $f \in \Psi_i$ and $f \in \Psi_j$, for $i < j$ such that node r_i lies on the path from the root to node r_j . If p lies below the support hyperplane of f , then either $p \in \sigma_k$ for $j < k$, or $p \notin \Sigma^d$.*

Using the Hole theorems, a modified point/complex classification algorithm for the compress spatial index can be stated.

Algorithm PC2(Point/Complex Classification) Let r_m be the root node of the compressed spatial index for Σ^d and p the point to be classified. Return **out** if the point is outside the complex, or return the k -face of Σ^d that contains p , for the minimum k .

1. If r_m represents **out** then return it and skip the remaining steps.

2. If r_m represents a k -face, then return $[r_m, \emptyset, -\infty, +\infty]$, and skip the remaining steps.
3. Let $[l, u]$ be the current bounds. If $u \leq m$ then continue from Step 1 with the *below* child node of r_m . If $m \leq l$ then continue from Step 1 with the *above* child node of r_m .
4. Let Ψ_m be the separation sheet of node r_m . Recursively obtain the faces $[f, f', l', u']$ where $f \in \Psi_m$ for $\Pi_d(p) \in \Pi_d(f)$, and if p projects onto a k -face, f' is adjacent to f for $k > d - 1$.
5. Compare p against the hyperplane of f and proceed depending on whether p is above, on or below as follows:

Above Classify p starting from Step 1 with the *above* child node of r_m and the bounds $[\max(l, l'), \min(u, u')]$.

On Return $[\sigma_i, f', l', u']$ if $m \geq 1$ with $\sigma_{\pi_{l'}}$ and $\sigma_{\pi_{u'}}$ being the adjacent simplices below and above adjacent to σ_i , and with $i = m$ if $m > 0$, or $i = 1$ if $m = 0$.

Below Classify p starting from Step 1 with the *below* child node of r_m and the bounds $[\max(l, l'), \min(u, u')]$.

Algorithm PC2 is identical to PC1 except for the additional bounds that are returned and checked during traversal. The returned bounds on occasion direct the search to bypass the recursive search of a separation sheets thus reducing the number of subtrees that need to be searched. In the worst case, however, no reduction may occur and the the full height of the tree needs to be searched throught all dimensions requiring the same $O(\log^d dn)$ time as for Algorithm PC1.

5 Discussion of the Technique

It is useful to compare the spatial index presented in this paper with the another multi-dimensional volume-based data structure used in solid modeling. The spatial index presented here has been motivated in part by the brep-index of Vaněček [Van90b] with which we will compare the spatial index. The brep-index is a generalization of the binary space partition (BSP) trees [FKN80, Van90a], and motivated indirectly by the cut trees of Dobkin and Edelsbrunner [DE87].

Figure 8: An example of a brep-index constructed for the top face of the solid shown on the top left.

In the brep-index, the boundary of an object, given by a boundary representation (brep), is recursively cut by oriented (splitting) planes, thus fragmenting the boundary into pieces that do not cross the splitting planes. This partitions space into convex regions that can be arranged into a multi-dimensional ternary-tree data-structure. Each region is either completely inside or outside the object, or lies inside a region spanned by some vertex, edge or face of the object. Figure 8 shows an object, its top face fragment by the splitting planes and the corresponding brep-index.

Although a brep-index can be constructed for a simplicial complex, the storage requirement for maintaining the fragmentation of each k -simplex would be prohibitive. The spatial index, on the other hand, is a structure that does not fragment the simplicial complex; it partitions it.

The disadvantage of the spatial index method is that the use of the separation sheets imposes a convexity constraint on the simplicial complex. Since the brep-index decomposes along the boundary of an object, the objects can be concave

with an arbitrary number of through holes.

Another comparison between the two approaches lies in the search strategies. The brep-index is a top-down search structure, while the spatial index is a bottom-up search structure. In the brep-index, a point/object classification follows a single path from the root to a leaf and terminates with the answer. This has the advantage of being robust in that a tolerance for determining when the point lies on the boundary can be changes as necessary without resulting in contradictory results. In the spatial index, the search begins in zero-dimensional space and moves up into higher dimension. This imposes checking the projections of the query point rather than the point itself in the highest dimensional space thereby causing uncertainty as to the correct k -face containing the point. We are currently addressing the question of robustness for the spatial index, since the useful applications of the spatial index rest on a robust implementation.

In summary, we note that either data structure has certain appealing properties while not being totally without problems. We are attempting to devise a unified representation that exhibits the appealing properties of both while avoiding their problems.

We consider one additional observation. Let P^d be the regular polyhedron of dimension d that is the support space of Σ^d , where the *support space* of Σ^d is the set union of simplices in Σ^d . An extrusion operation is a simple technique for the generation of higher dimensional polyhedra P^d (see [FP90]). If $P^d \subset E^d$, and $t \in E^{d+1}$, the linear extrusion is defined as $S_t(P^d) = P^d \times T$, where $T = [0, t]$. If a winged representation of P^d is given, and the number of d -simplices in the representation is n , a winged representation of $S_t(P^d)$ can be computed in $O(d^2n)$ by the algorithm described in [FP90]. For this particular class of convex polyhedra a useful property holds, namely, if Ψ is a separation sheet for P^d , then $S_t(\Psi)$ is a separation sheet for P^{d+1} . This circumstance implies that, once the spatial index for P^d is available, the spatial index for P^{d+1} can be constructed with an “ad hoc” technique of using extrusion for computing the separation sheets.

In conclusion, we state the open problems. How can the spatial index be extended to nonconvex simplicial complexes? The basic problem that arises is that by our definition there may not be a single lower envelope that spans the projection of the simplicial complex. This is illustrated in Figure 9. One approach is to first take the convex hull of the simplicial complex, and triangulate the concavities. This would insert simplices and their faces would be used in computing the separation sheets. But, as far as we know, there are no general algorithms for this in higher dimensions. However, if a nonconvex simplicial complex yields

Figure 9: An example of a simplicial complex for which there is no lower envelope in either of the two possible projections.

monotone separation sheets, our algorithm holds. There is nevertheless no practical way of knowing a priori whether a nonconvex simplicial complex has this property.

Acknowledgements

We wish to thank Christoph Hoffmann for introducing us and giving us the opportunity to work together on this common problem, and for guiding our approach. Also, we like to thank Franco Preparata for discussing with us his chain approach and indirectly supporting this work.

References

- [AHU83] A. V. Aho and J. E. Hopcroft and J. D. Ullman *Data Structures and Algorithms*, Addison-Wesley, 1983.
- [Ber90] F. Bernardini, "Integration of Polynomials over n -dimensional Polyhedra" to appear in Computer Aided Design, Special Issue *Beyond Solid Modeling*, February 1991.
- [DE87] D. P. Dobkin and H. Edelsbrunner, "Space Searching for Intersecting Objects." *Journal of Algorithms*, 8:348-361, 1987.
- [Fer90] V. Ferrucci, "*Simple_x*: User Manual and Implementation Notes, Part1," *Università di Roma "La Sapienza", Dip. di Informatica e Sistemistica*, Tech. Rep. 06-90, Rome, May 1990.

- [FP90] V. Ferrucci and A. Paoluzzi, "Extrusion and Boundary Evaluation for Multidimensional Polyhedra", to appear in Computer Aided Design, Special Issue *Beyond Solid Modeling*, February 1991.
- [FKN80] H. Fuchs and Z. M. Kedem and B. F. Naylor, "On Visible Surface Generation by a Priori Tree Structures." *Conf. Proc. of SIGGRAPH '80*, 14(3):124-133, July 1980.
- [LP77] D. T. Lee and F. P. Preparata, "Location of a point in a planar subdivision and its applications," *SIAM Journal on Computing*, 6(3):594-606, Sept, 1977.
- [PC90] A. Paoluzzi and C. Cattani, "Simplicial-Based Representation and Algorithms for Multidimensional Polyhedra", manuscript, 1990, revised and extended version of the paper: C. Cattani and A. Paoluzzi, "Solid Modeling in Any Dimension". *Dip. di Informatica e Sistemistica*, Università "La Sapienza", Techn. rep. 02-89, Rome, Jan 1989.
- [PS86] L. K. Putnam and P. A. Subrahmanyam, "Boolean Operations on n-Dimensional Objects." *IEEE Comp. Graph. and Appl.*, 6(6):43-51, 1986.
- [PS85] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. Springer-Verlag, 1985.
- [ROC89] J. R. Rossignac and M. A. O'Connor, "SGC: A dimension-independent model for point sets with internal structures and incomplete boundaries." *Geometric Modeling for Product Engineering, Proceedings of the 1988 IFIP/INSF Workshop on Geometric Modeling, Rensselaerville, NY*, Sept. 18-22, 1988. M. Wozny, J. Turner and K. Press, eds., North Holland, 1989, pp. 145-180.
- [Van90a] G. Vaněček, Jr, "Towards Automatic Grid Generation using Binary Space Partition Trees," *Purdue University, Department of Computer Science*, CAPO Report CER-90-6, West Lafayette, IN, Jan 1990.
- [Van90b] G. Vaněček, Jr, "A Data Structure for Analyzing Collisions of Moving Objects," *Hawaii International Conference on System Science, HICSS-24*, Kuai Hawaii, January 1991.

Figure 10: Example complex with one simplex for Appendix A.

Figure 11: The full spatial index for the simplicial complex of Figure 10. Each k -face is indicated by its vertices (in bold), and the equation of the hyperplane.

A Simple Example of a Full Spatial Index

We now show a simple, but nontrivial example in 2D. The complex consists of a single 2-simplex shown in Figure 10. The spatial index is shown in Figure 11, where the k -faces are indicated by the sign and the vertex indices of the k -face. The complete data structure contains six internal nodes, and fifteen face nodes.