[9] Hiroshi Inagaki, Kokichi Sugihara, and Noboru Sugie. Numerically robust incremental algorithm for constructing three-dimensional voronoi diagrams. In *Proc. 4th Canad. Conf. Comput. Geom.*, pages 334–339, 1992.

[10] D. G. Kirkpatrick. Efficient computation of continuous skeletons. In *Proc. 20th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 18–27, 1979.

[11] Jean-Claude Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, Boston, USA, 1991.

[12] David Lavender, Adrian Bowyer, James Davenport, Andrew Wallis, and John Woodwark. Voronoi diagrams of set-theoretic solid models. *IEEE Comput. Graph. Appl.*, 12(5):69–77, 1992.

[13] D. Leven and M. Sharir. Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams. *Discrete Comput. Geom.*, 2:9–31, 1987.

[14] A. Lingas. Fast algorithms for bounded Voronoi diagrams of restricted polygons. In *Proc. 2nd Canad. Conf. Comput. Geom.*, pages 204–208, 1990.

[15] T. Lozano-Pérez. Spatial planning: a configuration space approach. *IEEE Trans. Comput.*, 32:108–120, 1983.

[16] C. Ó'Dúnlaing, M. Sharir, and C. K. Yap. Retraction: a new approach to motion-planning. In *Proc. 15th Annu. ACM Sympos. Theory Comput.*, pages 207–220, 1983.

[17] C. Ó'Dúnlaing and C. K. Yap. A "retraction" method for planning the motion of a disk. *J. Algorithms*, 6:104–111, 1985.

[18] Atsuyuki Okabe, Barry Boots, and Kokichki Sugihara. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams.* John Wiley & Sons, Chichester, England, 1992.

[19] D. Siersma. Private communication, 1995.

[20] K. Sugihara and M. Iri. Construction of the Voronoi diagram for 'one million' generators in single-precision arithmetic. *Proc. IEEE*, 80:1471–1484, 1992.

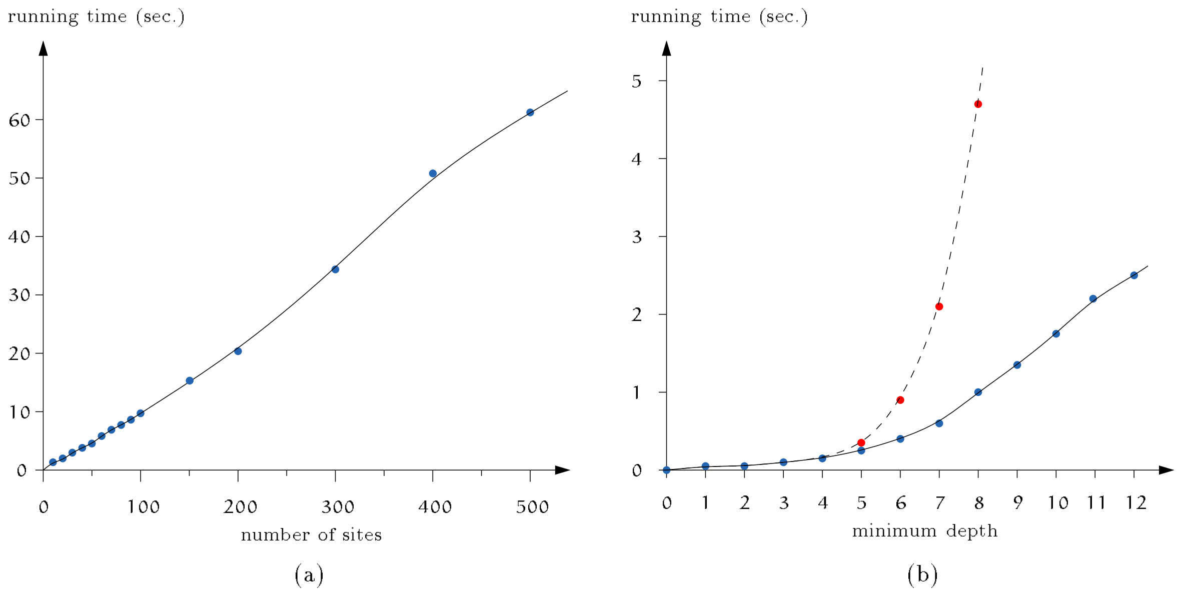

Figure 5: The running time set against (a) the number of sites, and (b) the recursion depth.

robots (that is, robots that are allowed to rotate) and if it can be used to efficiently compute the dual of the Voronoi diagram, the *Delaunay triangulation*. Finally, can the algorithm be modified in such a way that favorable asymptotic complexity bounds can be proven?

References

[1] H. Alt and C. K. Yap. Algorithmic aspect of motion planning: a tutorial, part 2. *Algorithms Rev.*, 1(2):61–77, 1990.

[2] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor search. In *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pages 573–582, 1994.

[3] F. Aurenhammer. Voronoi diagrams: a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23:345–405, 1991.

[4] Christoph Burnikel, Kurt Mehlhorn, and Stefan Schirra. How to compute the voronoi diagram of line segments: Theoretical and experimental results. In *Proc. 2nd Annu. Sympos.*, volume 855 of *Lecture Notes in Computer Science*, pages 227–237, 1994.

[5] L. P. Chew. Building Voronoi diagrams for convex polygons in linear expected time. Technical Report PCS-TR90-147, Dept. Math. Comput. Sci., Dartmouth College, Hanover, NH, 1986.

[6] L. P. Chew and R. L. Drysdale, III. Voronoi diagrams based on convex distance functions. In *Proc. 1st Annu. ACM Sympos. Comput. Geom.*, pages 235–244, 1985.

[7] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 7:381–413, 1992.

[8] J. E. Hopcroft, J. T. Schwartz, and M. Sharir. *Planning, Geometry, and Complexity of Robot Motion*. Ablex Publishing, Norwood, NJ, 1987.

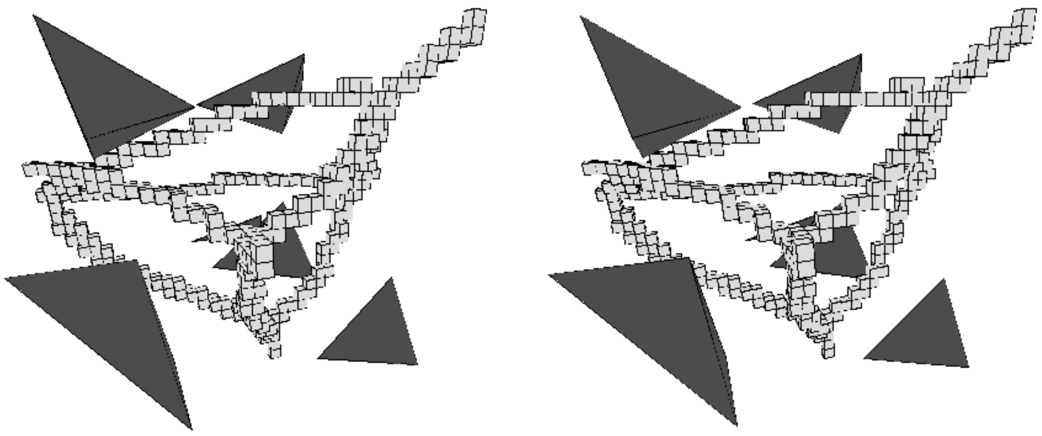

Figure 4: A stereogram view of the approximate diagram of a set of convex sites in $\mathbb{R}^3$.

(as described in Section 5). Figure 5b shows the results; the variable precision is indicated with a solid line. For small depths, the two curves coincide. After a certain point (approximately at a depth of five), the slope of the curve indicating the variable precision decreases because many cells of this size are empty, whereas the other curve's slope keeps increasing. Note that in this particular case the approximation will never be finished since the scene contains passages through which the robot cannot pass.

7 Conclusions and open problems

We described a method to efficiently approximate the Voronoi diagram of a set of disjoint convex sites in $\mathbb{R}^d$ within some predetermined precision, and its application to retraction motion planning. There are a number of advantages to this approach. The only primitive required is the computation of the distance to the nearest site from a given point. As a consequence, the method is very general and can be used for arbitrary convex sites. Implementing the algorithm is easy, and although the theoretical complexity of the algorithm is higher than existing approaches it is very efficient in terms of running time. Unlike many existing approaches it does not suffer from robustness problems, that is, it is insensitive to round-off errors introduced in the computation of the diagram, and it can handle degenerate diagrams without modifications. These properties make it well-suited for practical applications.

Although in this paper we have restricted ourselves to convex sites, we have a strong belief that the method works for concave sites as well. This setting would however require a much more elaborate study of the properties of both the Voronoi diagram and the approximate diagram, and a formal correctness proof seems difficult in this case. We furthermore believe that the method can be used to approximate the Voronoi diagram for a large class of metrics—in particular, for the Minkowski metrics L_m. Some of the lemmas would have to be revised to account for this; for arbitrary metrics, some of the properties of the approximate diagram will most likely be lost. It would be interesting to see if the method can be applied to free-flying

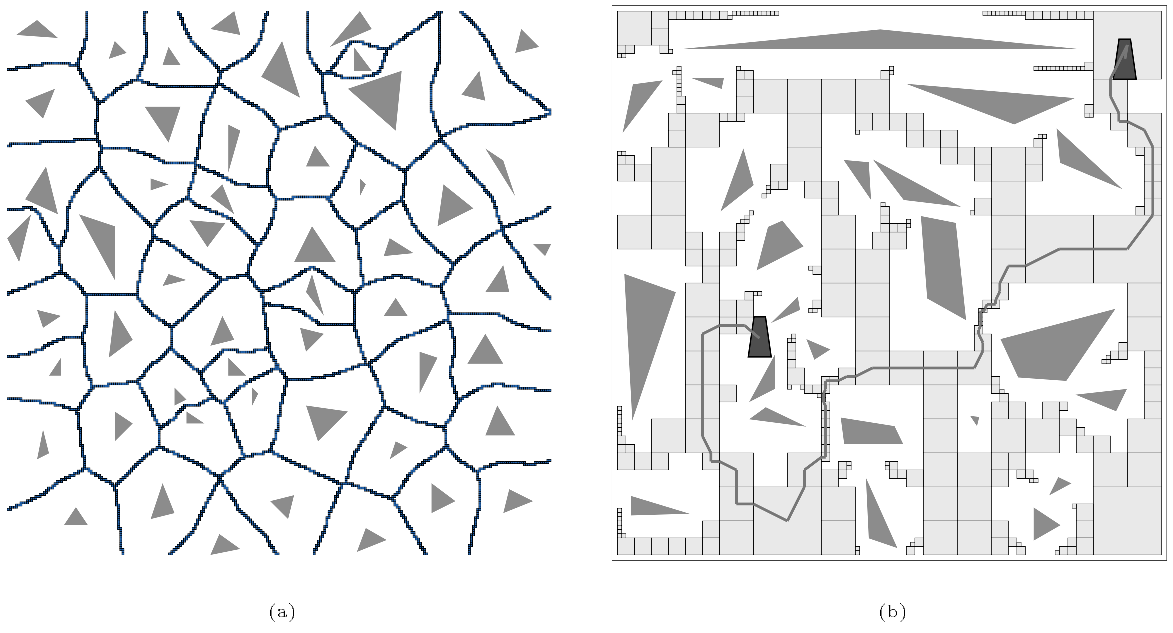

Figure 3: The approximate diagram (a) of a large number of sites, and (b) for a motion planning problem.

To gain insight into the practical complexity of the method, we tested the running time in relation to the number of sites. We did this by generating a total of n (with n varying from 10 to 500) disjoint triangular sites that together cover 20% of the unit square, and approximating the Voronoi diagram to a depth of eight (that is, cells of size $1/256$). Figure 5a shows the running time of the program set against the number of sites. The running time of the algorithm is expected to be proportional to the number of obtained cells times the number of sites, since for each vertex of a cell we compute the distance from each obstacle. A possible way to improve this is to keep track of 'interesting' sites for each cell C in the recursion, by ruling out those sites that can not be closest to a vertex of any cell that is created by subdividing C. That way we need to compute distances to (asymptotically) only $O(n/4^k)$ sites for each vertex, where k is the recursion depth. This could also be used to obtain a better theoretical complexity of the algorithm.

Figure 5a seems to indicate a linear behavior when n increases; this suggests that the number of cells does not increase. This though is not true: the number of cells for $n = 200$ is about five times as large as for $n = 10$. The reason probably is that sites that are far away take little time to test because a simple bounding box test suffices. Hence, the time bounds are dominated by the time required to compute the distance to nearby sites. As noted before, we could improve the running time by keeping track of 'interesting' sites while recursively subdividing the space. The above observation though suggests that in practice this will not help much because we would only loose the 'easy' sites.

We also tested the running time in relation to the precision of the approximate diagram, by approximating the diagram of Figure 3b with increasing precision in two ways: first we approximated the diagram with a *fixed precision*, that is, with cells of equal size; next we approximated the diagram with *variable precision* by not subdividing empty cells any further

Proof. The minimum clearance along P is achieved at a point p closest to the boundary of C. Since every cell C' adjacent to C is at least the size of C, the portion of the skeleton within C is located at a constant distance of size$(\mathsf{C})/2$ from the boundary of C. □

Although this approach resembles the standard approximate cell decomposition [11], there is an important difference: whereas the standard cell decomposition subdivides every cell that is not either *empty* or *full*, our approach refines only those cells that intersect the Voronoi diagram as well. Consequently, the number of cell to be subdivided will be smaller, and the resulting diagram will contain less redundancy. For example, standard approximate cell decomposition approximates the boundary of every obstacle with very small cells, whereas our approach only does this at places where two obstacles are very close.

As mentioned before, an important property of our method is that it is robust. It is not only insensitive to round-off errors introduced in the computation (as discussed in Section 4), even inaccurate information does not cause any problems. For example, a cell can slightly intersect an obstacle but accidentally be labeled as located entirely in CF. Because the skeleton is defined in such a way that it does not come close to the cell boundaries (except where two cells are adjacent—this obviously does not cause a problem) the robot stays well away from the obstacles.

6 Experimental results

We implemented the algorithm described in the previous sections in C++ on a Silicon Graphics Indigo II workstation, which is based on an R4400 processor running at 150 MHz and rated on the SpecMarks benchmark with 93 SPECfp92 and 85 SPECint92. The program computes the approximate diagram of a set of convex polygons and also plans the translational motion of either a point or a convex polygon. It comprises approximately 1700 lines of source code, 500 of which contain the implementation of Algorithm Hierarchical; the rest is used for the graphical interface, user interaction, and file handling. To demonstrate the efficiency of the algorithm we now provide some experimental data obtained with the program.

Figure 3 shows the approximate diagram of a large number of sites and the approximate diagram for a motion planning application. The diagram of Figure 3a was computed in about four seconds; the diagram was approximated to a depth of eight in order to obtain an accurate approximation. In Figure 3b, a robot moves from the middle left to the top right amidst a set of obstacles of various sizes. The approximate diagram was computed in less than one second; a path for the robot is indicated. Note that the diagram contains some large cells that are not subdivided because they can not cause the robot to collide with an obstacle, whereas the precision is locally increased where necessary; this speeds up the computation considerably. These results indicate that the efficiency of our approach is comparable to existing algorithms.

To show that the same algorithm works for arbitrarily shaped convex sites, we modified the aforementioned implementation to work for a set of discs in the plane. The diagram shown in Figure 1 was computed in less than five seconds using this implementation; the depth of the approximation was ten. We also have an implementation to compute the approximate diagram of a set of convex polytopes in $\mathbb{R}^3$; Figure 4 shows a stereogram of (the skeleton of) such a diagram, which was computed in just under a minute. We expect that this implementation can be optimized in a number of ways—in particular, the distance computations are currently done in a not so efficient way.

Proof.
$$\begin{aligned} d(x, A_i^*) &= d(x, A_i \ominus P) \\ &= \inf\{d(x,r) \mid r \in (A_i \ominus P)\} \\ &= \inf\{d(x, q-p) \mid p \in P, q \in A_i\} \\ &= \inf\{d(x+p, q) \mid p \in P, q \in A_i\} \\ &= d(P(x), A_i). \end{aligned}$$
□

This yields an easy way to compute the clearance that can be performed completely in the workspace. As in the case of a disc, we define a one-dimensional skeleton in this approximate diagram and search for a path along this skeleton.

While we assumed disjoint sites in the general framework, this is no longer possible in the motion planning setting since the configuration space obstacles can overlap even if the workspace obstacles are disjoint. Although this does not invalidate the method, we would waste a lot of effort in subdividing cells that are contained completely in a configuration space obstacle.

Lemma 5.4 *If* $P(v_i)$ *intersects an obstacle* A_j *for every vertex* v_i *of a cell* C*, then* C *is full, that is,* $C \subseteq CF^*$.

Proof. Suppose that $P(v_i)$ intersects A_j for each of the v_i. Because both P and A_j are convex, $P(v)$ also intersects A_j for any convex combination v of the v_i, which is just C. □

Using this test, we can avoid further subdivision of a cell that does not contain any configuration at which the robot can safely be placed. We now apply Algorithm HIERARCHICAL with the metric induced by clearance() to obtain the approximate diagram $\mathcal{V}_a$. The retraction is defined analogously to the case of a disc.

5.3 Improved computation of the diagram

The following observation can reduce the complexity of the approximate diagram. In the motion planning application it is not necessary to approximate the Voronoi diagram accurately; we merely need to construct a (topology-preserving) set of cells along which the robot can safely move.[2] Small cells are only required in small passages. Therefore we subdivide a cell *only if* it contains (part of) a configuration space obstacle, that is, if the L_∞ distance from the robot placed at the center of the cell to an obstacle is smaller than half the cell size. By adding this condition to Step 7 of Algorithm HIERARCHICAL we can avoid further subdivision of a cell if it is located entirely in CF. Note that by doing so we obtain an approximate diagram with cells of different sizes. Since this diagram contains the approximate diagram with equally sized cells, this does not affect the connectivity of the diagram. In this diagram the robot does no longer follow paths with maximum clearance since the cell boundaries can now be arbitrarily close to the obstacles. However, if we define a skeleton in the same way as before and move the robot only along the skeleton, we can guarantee a lower bound on the clearance along the diagram.

Lemma 5.5 *The clearance along a path* P *in* $\mathcal{V}_a$ *is at least* size$(C)/2$*, where* C *is a smallest cell of* P.

[2]It can still be desirable to keep the robot sufficiently far from the obstacles. There is however no need to approximate the whole diagram with high precision.

approximate diagram we can however approximate the Voronoi diagram arbitrarily close. We define the *depth* of a cell $C \in H$ as the $\log_{1/2}$ of its size, or equivalently, the number of times the unit hypercube has to be subdivided to obtain a cell of the size of C. The depth of the approximate diagram is given by the depth of a smallest cell.

Lemma 5.2 *If there exists a path P in $\mathcal{V}(\mathcal{A})$ such that the clearance along P is at least $\varepsilon > 0$, there exists a path in the skeleton of $\mathcal{V}_a(\mathcal{A})$ at a depth of $\lceil \log_{1/2} \varepsilon \rceil + 1$.*

Proof. Consider a point $p \in P$ and the open sphere D centered at p with radius ε. Since $\text{clearance}(p) \geqslant \varepsilon$, $D \cap \mathcal{A} = \emptyset$. Now any cell of size $\varepsilon/\sqrt{2}$ with p as an interior point is contained completely in D and thus is empty. This cell size is obtained at a depth of $\lceil \log_{1/2}(\varepsilon/\sqrt{2}) \rceil = \lceil \log_{1/2} \varepsilon \rceil + 1$. □

This lemma shows that there always exists a depth at which a path can be constructed; the depth depends on the minimum clearance along the Voronoi diagram. As a result, we can use the approximate diagram to plan the motion of a sphere.

5.2 Planning the motion of a polytope

The approach for a sphere can easily be generalized to the case of a convex polytope P by means of a *convex distance function* [6]. We define the clearance of P from A_i as

$$\text{clearance}(P(x), A_i) = \min\{d_P(x, y) \mid y \in A_i\} \tag{8}$$

where d_P is the convex distance function induced by P; for this it is necessary that P contains its origin as an interior point. The same diagram can again be used to plan the motion of any homothet of the original robot polytope [13]. However, it is not guaranteed that the robot stays far from the obstacles when travelling along the diagram—the minimum distance depends on the exact shape of the robot and the position of the origin in the robot polytope. This property is undesired if the method is to be used for real-world applications, since controlling a robot nearly always introduces some error in the trajectory. By maximizing the distance from the obstacles, we minimize the chance of the robot accidentally colliding with an obstacle. The clearance that realizes this is given by

$$\text{clearance}(P(x), A_i) = d(x, A_i^*). \tag{9}$$

This clearance gives the shortest distance that $P(x)$ has to be moved to intersect A_i; this is exactly the distance of x from the corresponding configuration space obstacle A_i^*. The Voronoi diagram under the metric induced by Equation (9) is the Euclidean Voronoi diagram generated by the configuration space obstacles, and thus maximizes the distance that P has to move to collide with the nearest obstacle. Unfortunately, we can not directly compute this because we do not have a description of the A_i^*. If $W \subset \mathbb{R}^2$, we can explicitly compute C by means of the *Minkowski difference*, given by $(P \ominus Q) = \{p - q \mid p \in P, q \in Q\}$ for polytopes P, Q. The configuration space obstacle generated by A_i is given by $A_i^* = A_i \ominus P$, and consequently $x \in CF$ if and only if $x \notin (\mathcal{A} \ominus P)$ [15]. While it is possible to compute this in the planar case, we prefer a different approach that computes the above clearance() in W instead.

Lemma 5.3 *The Euclidean distance in CF is equal to the clearance in W, that is, $d(x, A_i^*) = d(P(x), A_i)$.*

space obstacle A_i^* consisting of all configurations in which R collides with A_i. For given configurations $s, g \in CF$, we define a *motion* between s, g as a continuous map $f : [0, 1] \to CF$ such that $f(0) = s$, $f(1) = g$. The corresponding *path* is the set of configurations $f(t)$ for $t \in [0, 1]$.

The retraction method for motion planning [1, 8, 16, 17] uses the Voronoi diagram of the obstacles to compute a motion between given configurations s, g of the robot. First, a *retraction* of s, g onto configurations s', g' on the diagram is computed, and next s', g' are connected through a path entirely on the diagram. Because points on the diagram are equidistant from the d closest obstacles, the resulting paths have the nice property of maximizing the clearance of the robot.

5.1 Planning the motion of a disc

Ó'Dúnlaing and Yap [17] showed that this method can be used to plan the (planar) motion of a disc. Each configuration is retracted onto the Voronoi diagram of the obstacles by moving away from the nearest obstacle until we reach the Voronoi diagram. A disc can be moved along a path on the diagram if the minimum clearance along this path is greater than its radius. Thus the same diagram can be used to plan the motion of discs with different radii.

We follow the same method with our approximate diagram. Using Algorithm HIERARCHICAL, we first compute the approximate diagram of the obstacles. Note that in the motion planning application we only want to use those cells of the approximate diagram that do not contain (part of) an obstacle, because the robot can not collide with an obstacle when moving inside these cells. We call a cell C *empty* if it does not contain (part of) an obstacle; this can be checked by means of the *convex distance function* d_C induced by C [6] as follows. By definition of a convex distance function, a convex polytope P that includes the origin as an interior point intersects a polyhedron Q if and only if $d_P(o, Q) \leqslant 1$. In our case we can alternatively use the d_{L_∞} metric to determine whether a cell is empty, as shown by the following lemma.

Lemma 5.1 *Let C be a cell centered at c, then C intersects an obstacle A_i if and only if $d_{L_\infty}(c, A_i) > \mathrm{size}(C)/2$.*

Proof. The set of points at unit distance from a point p under d_{L_∞} is just the axis-parallel hypercube with sides of length 2 and centered at p; hence, $d_{L_\infty}(c, A_i) > \mathrm{size}(C)/2 \Leftrightarrow d_C(c, A_i) > 1$. ◻

We add this condition to Line 6 of Algorithm HIERARCHICAL. Next, we define a one-dimensional skeleton in $\mathcal{V}_a$ as follows. Let C_1, C_2 be adjacent cells of $\mathcal{V}_a$, c_1, c_2 the centers of C_1, C_2, respectively, h the side of C_1 which is adjacent to C_2, and x the midpoint of h; we call x the *via point* of C_1, C_2. We connect C_1 and C_2 through the line segments $\overline{c_1x}$ and $\overline{xc_2}$. The resulting skeleton clearly preserves the connectivity of $\mathcal{V}_a$ and thus is connected if $\mathcal{V}_a$ is connected. Each configuration x is retracted onto the approximate diagram by moving away from the nearest obstacle until we reach a cell C of the approximate diagram. This can not cause the sphere to collide with the obstacles because the clearance strictly increases during the move. Next we search for a path along the skeleton with sufficient clearance for the disc to move. Because of the approximate nature of the skeleton this is not always possible—even if there exists a path along the Voronoi diagram. By decreasing the size of the cells in the

```
Algorithm HIERARCHICAL
1.  $Q \leftarrow$ the unit hypercube
2.  $\mathcal{V}_a \leftarrow \emptyset$
3.  while $Q$ is not empty
4.      do $C \leftarrow$ an element of $Q$
5.          $Q \leftarrow Q \backslash \{C\}$
6.          if $|L(C)| > 1$
7.              then if $\text{size}(C) > \text{size}_{\max}$
8.                      then subdivide $C$ into $2^d$ smaller cells $C_i'$
9.                           $Q \leftarrow Q \cup \{C_i'\}$
10.                          subdivide cells adjacent to $C$ and of size $> \text{size}(C)$
11.                     else $\mathcal{V}_a \leftarrow \mathcal{V}_a \cup \{C\}$
12. return $\mathcal{V}_a$
```

Table 2: Hierarchical computation of the diagram.

This algorithm gives an efficient way to construct the approximate Voronoi diagram; some experimental results are presented in Section 6.

An advantage of the method proposed here is that it is robust. In particular, round-off errors introduced in the computation of the diagram do not cause any problems if we can guarantee that coinciding vertices of adjacent cells are labeled identically—even with the possibility of round-off errors occurring in distance computations. This can easily be satisfied by computing the label of a vertex only once and storing references to this label at the cells sharing this vertex. Furthermore, degenerate diagrams are automatically handled correctly without any requirement for special cases.

5 Application to motion planning

In this section we describe the application of the general framework of Section 3 to a classical problem that can be solved by means of the generalized Voronoi diagram: motion planning using retraction. We demonstrate the efficiency of the resulting algorithm through experimental results.

The objective of the *motion planning problem* is finding a path for a robot R moving from a source to a goal position amidst a set of obstacles. The workspace W of the robot is a closed subset of $\mathbb{R}^d$ that contains a set of n obstacles $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$. We assume that $\mathcal{A}$ is finite and that each obstacle consists of a simple convex polytope including its interior; by abuse of notation, we also denote the union of the obstacles by $\mathcal{A}$. The area outside W is also regarded as an obstacle. By fixing a reference point to the robot R we can describe any placement of R by the coordinates of its reference point in W, assuming we only allow for translations; we call such a specification a *configuration*. The robot placed at a configuration x is denoted by $R(x)$. The configuration space C is the space consisting of all possible configurations of R in W and is a closed subset of $\mathbb{R}^d$. For simplicity, we assume C to be normalized to $[0,1]^d$. The *free configuration space* CF consists of all configurations for which $R(c)$ does not collide with $\mathcal{A}$. Similarly, the *forbidden configuration space* $CF^* = C \backslash CF$ is the subset of C in which R collides with $\mathcal{A}$. Every obstacle A_i defines a *configuration*

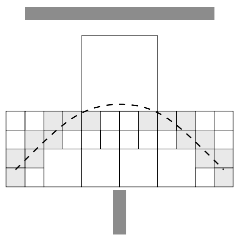

Figure 2: A disconnected approximate bisector.

cells that are adjacent to cells on the approximate diagram but are not part of the diagram themselves—in fact, we examine every such cell.

To avoid this we instead start with a coarse approximation of the diagram and locally refine it, discarding parts that are of no further interest. This process is repeated recursively until we end up with cells of the desired size. Initially, the approximate diagram $\mathcal{V}_a$ is empty. Let C be the unit hypercube, and perform the following algorithm on C. First, determine the labels (which have not yet been computed) of the vertices v_i of C as given by Equation (5). If not all l_i are identical, Lemma 3.2 states that C intersects the Voronoi diagram. Let size_{max} denote the requested size of the cells in the approximate diagram. If the size of C is size_{max} we add C to $\mathcal{V}_a$; if C is larger than size_{max}, we locally refine the approximation by subdividing C into a set of smaller cells as follows. Bisect the faces of C with d hyperplanes perpendicular to the coordinate axes as described in Section 3. The arrangement of the hyperplanes within C consists of 2^d cells $C'_1, C'_2, \ldots, C'_{2^d}$. Discard C and recursively apply the procedure to the C'_i. Although one would expect that this algorithm correctly computes the approximate diagram, there are cases in which this approach fails; see Figure 2 for an example in the plane. The top cell has identical labels and thus is not subdivided even though it intersects the Voronoi diagram, and as a result the diagram becomes disconnected. To overcome this problem, we subdivide any cell adjacent to C and at least twice the size of C, and add the resulting cells to the recursion. Even with this additional step the resulting diagram does not include all cells of the approximate diagram—we can however show that the diagrams are equivalent. The resulting algorithm is shown in Figure 2.

Lemma 4.1 *The diagram computed by Algorithm* HIERARCHICAL *is connected if the Voronoi diagram is connected.*

Proof. We will show that cells of the approximate diagram that are not found by the algorithm do not disconnect the resulting diagram. Let $C \in \mathcal{V}_a(\mathcal{A})$—in other words, $L(C) > 1$ and $\text{size}(C) = \text{size}_{\text{max}}$—and suppose that C is not found by the algorithm. This can happen only if some larger cell C' containing C has identical labels and thus is not subdivided by the algorithm. Furthermore, the size of the cells adjacent to C' is at most half that of C', otherwise C' would have been subdivided in Step 10 of the algorithm. Since the Voronoi diagram intersects C by Lemma 3.2, it also intersects C'. According to Corollary 3.6, either $L(C') > 1$—a contradiction—or C' is (in the diagram with cells the size of C') adjacent to a cell C'' with $L(C'') > 1$ and $\text{size}(C'') = \text{size}(C')$. Since $\text{size}(C'') \geqslant \text{size}(C')/2$, C'' is subdivided at most once, and the approximate diagram inside C'' is connected. ◻

As a result, the approximate diagram can be used for various problems that traditionally require the computation of the Voronoi diagram. One such application is the *post-office problem*, which asks for the nearest site from a given query point. The approximate diagram can be used to provide an approximate nearest neighbor; Arya et al. [2] describe another approximate solution to this problem. A second application is the *motion planning problem*, which consists of finding a collision-free motion for a robot moving amidst a set of obstacles. We elaborate on the latter in Section 5.

4 Construction of the approximate diagram

A straightforward way to compute the approximate diagram defined in the previous section is by subdividing the unit hypercube into a set H of cells (as described in Section 3) and determining the labels of every cell in H. By definition, the approximate diagram consists of the cells in H with at least two different labels. However, this method wastes a lot of effort in computing labels of cells that are not important because they are located far from the Voronoi diagram and, hence, can not contribute to the approximate diagram.

A better way to determine which cells in H define the approximate diagram is the following. Suppose we know a single cell C that is part of the diagram; such a cell can easily be determined (see the *retraction* defined in Section 5). We mark C, and determine for every unmarked cell C' adjacent to C whether $L(C') > 1$. If this is the case, C' is also part of the approximate diagram, and the process is repeated recursively for C'; otherwise we mark and discard C'. The resulting algorithm is shown in Figure 1. Its correctness follows from the

```
Algorithm TRACING
1.  $C \leftarrow$ a cell of the approximate diagram
2.  $Q \leftarrow \mathcal{V}_a \leftarrow \{C\}$, and mark $C$
3.  while $Q$ is not empty
4.     do $C \leftarrow$ an element of $Q$, and mark $C$
5.        $Q \leftarrow Q \setminus \{C\}$
6.        for all unmarked cells $C'$ adjacent to $C$
7.            do if $|L(C')| > 1$
8.                  then $Q \leftarrow Q \cup \{C'\}$
9.                       $\mathcal{V}_a \leftarrow \mathcal{V}_a \cup \{C'\}$
10.                 else mark $C'$
11.  return $\mathcal{V}_a$
```

Table 1: Tracing the approximate diagram.

fact that the approximate diagram is connected (for a sufficiently small cell size). Note that the test in line 7 can be done using at most 2^d nearest neighbor queries under the Euclidean metric to determine the labels of the vertices. Double calculations of labels for the same vertex can easily be avoided by caching the results. Furthermore, as soon as we find that two vertices have different labels we can conclude that the cell is intersected by the Voronoi diagram; the remaining labels need not be determined. As an optimization, for this test we first check the labels that have already been computed. Thus the number of cells tested is in the same order as the size of the approximate diagram. Still we waste a lot of time looking at

This lemma shows that, in the case of the Euclidean diagram of a set of point sites, the approximate diagram covers the Voronoi diagram. We can approximate the diagram as closely as we wish by choosing the cells sufficiently small.

When the sites are convex objects, the Voronoi diagram is not necessarily covered by the approximate diagram. If however we choose the cells sufficiently small, the approximate diagram is connected if the Voronoi diagram is connected. The following lemma due to Siersma [19] plays a crucial role in this.

Lemma 3.5 *The bisector of two convex sites in $\mathbb{R}^d$ is differentiable with a bounded derivative.*

Although this result seems straightforward, it requires a rather lengthy proof. By choosing the cells sufficiently small, the part of the diagram intersecting a given cell has an arbitrarily small curvature—in other words, it approximates a straight line segment arbitrarily precise. The following is a direct consequence of this.

Corollary 3.6 *For sufficiently small cells, let C be a cell whose interior intersects a bisector* $\mathrm{bis}()$. *Then either $L(C) > 1$ or $L(C') > 1$, where C' is a cell adjacent to C.*

As a result, if we choose the cells sufficiently small, each cell that intersects a bisector is at most one cell 'off' the approximate bisector. As in the case in which every bisector is a hyperplane, the same holds for the complete diagram.

Lemma 3.7 *For sufficiently small cells, let C be a cell whose interior intersects the Voronoi diagram. Then either $L(C) > 1$ or $L(C') > 1$, where C' is a cell adjacent to C.*

Proof. Analogous to the proof to Lemma 3.4. ◻

Since every cell that intersects the Voronoi diagram is at most one cell 'off' the approximate diagram, it approximates the Voronoi diagram with a good precision.

Corollary 3.8 *For cells of size ε (ε sufficiently small), a point on the approximate diagram is at most $\varepsilon\sqrt{d}$ from the Voronoi diagram, and vice versa.*

Proof. Since the real diagram intersects every cell of the approximate diagram, a point on the approximate diagram is at most $\varepsilon\sqrt{d}$ from the real diagram. Conversely, Lemma 3.7 implies that the approximate diagram is at most one cell off the cells that are intersected by the real diagram; a point on the diagram is consequently at most ϵ from the nearest point on the approximate diagram. ◻

This shows that by decreasing the size of the cells in H we can approximate the Voronoi diagram arbitrarily precise. We summarize the main properties of the approximate diagram in the following theorem.

Theorem 3.9 *For sufficiently small cells, the approximate Voronoi diagram is connected if the Voronoi diagram is connected and can be used to approximate the Voronoi diagram with arbitrary precision.*

Definition 3.1 *The approximate bisector of two sites is defined as*

$$\mathrm{bis}_a(A_i, A_j) = \{C \in H \mid \{i, j\} \subseteq L(C)\} \tag{6}$$

that is, a cell is part of some approximate bisector if and only if not all its labels are identical. On the other hand we call the set

$$V_a(A_i) = \{C \in H \mid L(C) = \{i\}\} \tag{7}$$

the approximate Voronoi region of A_i. *The approximate Voronoi diagram* $\mathcal{V}_a(\mathcal{A})$ *generated by* $\mathcal{A}$ *is the set of cells in* H *with non-identical labels. In addition, we define the* $(d-k)$*-dimensional* approximate Voronoi faces *as the set of cells* $\{C \in H \mid |L(C)| > k\}$.

It is clear that every cell in H is part of either an approximate region or the approximate diagram; together they cover the unit hypercube. The approximate diagram consists of the cells whose vertices are located in different Voronoi regions.

3.2 Properties of the approximate Voronoi diagram

Let C be a cell of the subdivision H. The claim is that the labels of the vertices of C, as defined by Equation (5), can be used to determine whether C intersects the Voronoi diagram.

Lemma 3.2 *A cell* C *intersects the Voronoi diagram if* $L(C) > 1$.

Proof. Assume that $L(C) > 1$, and let v_i, v_j be vertices of C with different labels. The line segment $\overline{v_i v_j}$ intersects the bisector of the corresponding sites at some point p. Since C is convex, $p \in C$. ◻

This lemma shows that each cell of the diagram is intersected by the real diagram. The reverse is not necessarily true; sometimes the Voronoi diagram might run through cells that do not belong to the approximate diagram. Only for particular diagrams this is not the case.

Lemma 3.3 *If a bisector* bis() *is a* $(d-1)$*-dimensional hyperplane, the interior of a cell* C *intersects* bis() *if and only if* $L(C) > 1$.

Proof. The 'if' part follows from the previous lemma. Now suppose that bis() intersects int C. Since C is convex, bis() separates at least one vertex of C from the other vertices; therefore C's labels cannot be identical. ◻

It follows that a single bisector is covered by its approximate counterpart. The same holds if we combine a number of bisectors into a Voronoi diagram:

Lemma 3.4 *If every bisector of two sites forms a* $(d-1)$*-dimensional hyperplane, the interior of a cell* C *intersects the Voronoi diagram if and only if* $L(C) > 1$.

Proof. The 'if' part follows from Lemma 3.2. Now let C be a convex polytope intersecting the Voronoi diagram, v a vertex of C with label i, and $\mathrm{bis}(A_i, A_j)$ a bisector intersected by C. By the previous lemma, $L(C) > 1$ in the absence of other bisectors; let w be a vertex of C that is (in the absence of other sites) not located in $\mathrm{dom}(A_i)$. Since $\mathrm{dom}(A_i)$ is the intersection of the dominance regions of A_i over all other sites, w cannot be located in $\mathrm{dom}(A_i)$ if we add other sites (and bisectors). It follows that $L(C) > 1$. ◻

equal distance from the two closest sites and divide the space into the (open) Voronoi regions. The $(d-1)$-dimensional faces meet in $(d-2)$-*dimensional Voronoi faces* that achieve equal distance from the three closest sites. Finally, the 2-dimensional faces meet in 1-dimensional faces (that is, points) that achieve equal distance from the d closest sites. For convenience we refer to the $(d-1)$-dimensional faces as *Voronoi facets*, to the 2-dimensional faces as *Voronoi edges* and to the 1-dimensional facets as *Voronoi vertices*. The set of all Voronoi facets is called the *Voronoi diagram* $\mathcal{V}(\mathcal{A})$ generated by $\mathcal{A}$; the set of all Voronoi edges is called the *Voronoi skeleton*. In the following we use the terms region, face, etc. if it is clear that the *Voronoi* region, face, etc. is understood.

The requirement that the sites be disjoint is not really necessary, neither in the above nor in our method; it mainly serves to make the notions correspond more closely to their intuitive meaning. For example, if this requirement is relaxed the Voronoi facets includes those regions where two sites overlap. The method we are about to present works without any modification for overlapping sites.

3 The general framework

In this section we define a subdivision of the space into primitive cells, a subset of which forms an approximation to the Voronoi diagram.

3.1 The approximate Voronoi diagram

Let $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ be a set of n convex sites in $\mathbb{R}^d$ under the Euclidean metric. For simplicity, we scale the space such that the relevant portion of the Voronoi diagram of $\mathcal{A}$ is located inside the unit hypercube. To ensure that the diagram is connected, we treat the area outside the unit hypercube as an additional site. We now subdivide the unit hypercube into a set H of smaller hypercubes by means of $d(k-1)$ hyperplanes $h_{ij} : x_i = j/k$ for $1 \leqslant i \leqslant d$ and $1 \leqslant j \leqslant k-1$. That is, we construct $k-1$ hyperplanes normal to every coordinate axis and at equal distance $1/k$. The arrangement of these hyperplanes within the unit hypercube consists of a grid of k^d hypercubes of size $1/k$. Instead of axis-parallel hypercubes we could use cells of a different shape, such as simplices; however, hypercubes greatly simplify an implementation of the method. We call the hypercubes in H the *cells* of the approximation. Two cells are *adjacent* in H if their boundaries intersect in a $(d-1)$-dimensional hyperplane; a set S of cells is *connected* if every pair of cells in S is adjacent under transitivity.

Using the cells in H we define an approximate version of the Voronoi diagram generated by $\mathcal{A}$. Let C be a cell of H, and attach labels l_i to the vertices v_i of C such that[1]

$$\forall k \in [1,n] : \begin{cases} d(v_i, A_{l_i}) \leqslant d(v_i, A_k) & \text{if } l_i \leqslant k \\ d(v_i, A_{l_i}) < d(v_i, A_k) & \text{if } l_i > k \end{cases} \tag{5}$$

In other words, l_i indicates a unique site that is closest to v_i. Note that the label of v_i is uniquely defined even in the case in which v_i lies on a bisector: it is the site with the smallest index. We define $L(C)$ as the set $\{l_i \mid 1 \leqslant i \leqslant 2^d\}$ of labels of C, and more generally, $L(M)$ as the set of sites closest to the vertices of a polytope M.

[1]If no confusion is possible, we write v_i, l_i instead of $v_i(C), l_i(C)$.

sufficiently far away from the sites—in our approach the approximate bisector always stays close to the real bisector). Furthermore, in their paper no properties (like connectivity) of the approximate diagram are proven; such properties can be crucial for certain applications (for example, motion planning). Finally, the experimental results reported are quite bad. A simple two-dimensional diagram takes over an hour to compute, while our method solves a similar case in close to one second. (This is partially due to their inefficient implementation, but we believe that, due to their representation of the sites, an efficient implementation will still be considerably slower because their diagram consists of much more cells.)

The rest of this paper is organized as follows. In Section 2 we recall the definition of the Voronoi diagram and give some preliminary definitions. Next, in Sections 3 we define the approximate diagram and prove some important properties. In Section 4 we describe its efficient construction. We proceed with the application of the method to motion planning using retraction in Section 5 and provide some experimental results in Section 6. Finally, we conclude the paper in Section 7 where we also discuss some open problems and directions for possible future work.

2 Preliminaries

We first define the various terms and notations that will be used throughout the rest of the paper. Next we briefly review the Voronoi diagram.

The number of unique elements in a set S is denoted as $|S|$. Given a polytope M, we denote a copy of M placed at a position x as $M(x)$, and its vertices as $v_1(M), v_2(M), \ldots, v_m(M)$. For nonempty sets S, T of points, we define the distance of S from T under a metric d as

$$d(S,T) = \inf\{d(x,y) \mid x \in S, y \in T\} \tag{1}$$

In the case in which S consists of a single point x, we write $d(x,T)$ instead of $d(\{x\},T)$ to simplify the notation. Now let $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ be a set of disjoint sites in $\mathbb{R}^d$; by *site* we mean any convex compact set. The *bisector* of two sites under a distance function d is defined as

$$\mathrm{bis}(A_i, A_j) = \{p \mid d(p, A_i) = d(p, A_j)\} \tag{2}$$

that is, the locus of points at equal distance from both sites. Under the Euclidean metric the bisector of two sites is a curved surface; for some metrics however (for example, the Minkowski metrics L_1 and L_∞) the bisector can include a region [18]. The bisector divides the space into two regions and gives the *dominance region* of A_i over A_j, defined as

$$\mathrm{dom}(A_i, A_j) = \{p \mid d(p, A_i) \leqslant d(p, A_j)\} \tag{3}$$

Since the dominance region is closed, the dominance regions of two sites intersect in their bisector. We call the intersection of the dominance regions of A_i over all other sites, given by

$$V(A_i) = \bigcap_{1 \leqslant j \leqslant n, j \neq i} \mathrm{dom}(A_i, A_j) \tag{4}$$

the *(closed) Voronoi region* of A_i. The Voronoi region of A_i consists of all points having A_i as one of its closest sites; its boundary is composed of bisectors with other sites that we call $(d-1)$-*dimensional Voronoi faces.* That is, the faces consist of all points achieving

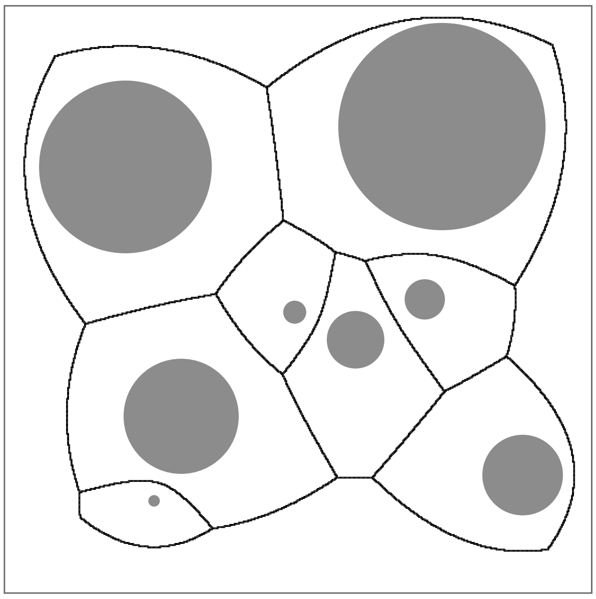

Figure 1: The approximate Voronoi diagram of a set of discs inside the unit square.

choose axis-parallel hypercubes because they are easy to handle in an actual implementation. The approximate diagram is defined in the same terms of bisectors of sites and regions as the ordinary Voronoi diagram. This framework can be constructed using the computation of the distance from a point to a site as the only primitive operation. As a result, the approach can be used for any set of sites for which such a distance function can be computed. For example, the method works well for discs or hyperspheres. To give an idea of what the approximate diagram looks like, Figure 1 shows the planar diagram of a set of discs under the Euclidean metric. Next, we describe an algorithm for the construction of the diagram. We start with a coarse approximation and locally refine it, discarding parts that are of no further interest. Finally, we apply the framework to a classical problem that can be solved by means of the Voronoi diagram: motion planning using retraction. Although theoretically the method is far from optimal, it turns out to run fast in practice.

The method presented here has several advantages that make it fit for practical purposes. Because of its generality, it can be used for a variety of applications. In addition, it is robust with respect to round-off errors introduced in the computation of the diagram, and it requires no special handling for degenerate cases. Furthermore, it is easy to implement due to its simplicity, and experiments show that the computation of the diagram is efficient. For example, the diagram shown in Figure 1 was computed in less than five seconds.

At this point we want to briefly discuss a related approach by Lavender et al. [12]. Their paper also describes a hierarchical approach to computing an approximate Voronoi diagram of a set of general sites in arbitrary dimension. However, their approach differs in a number of ways, making it (in our opinion) less useful. First of all, they represent the objects by an octree approximation. Secondly, they define the cells of the approximate diagram based on the distance between the cell and the approximate objects. As a result, the error made in the approximation becomes much larger and cannot be bounded (for example, for any cell size the approximate bisector of two point sites becomes arbitrarily wide when moving

Approximating Generalized Voronoi Diagrams in Any Dimension*

Jules Vleugels Mark Overmars

Abstract

Generalized Voronoi diagrams of objects are difficult to compute in a robust way, especially in higher dimensions. For a number of applications an approximation of the real diagram within some predetermined precision is sufficient. In this paper we study the computation of such approximate Voronoi diagrams. The emphasis is on practical applicability, therefore we are mainly concerned with fast (in terms of running time) computation, generality, robustness, and easy implementation, rather than optimal combinatorial and computational complexity. Given a set of disjoint convex sites in any dimension we describe a general algorithm that approximates their Voronoi diagram with arbitrary precision; the only primitive operation that is required is the computation of the distance from a point to a site. The method is illustrated by its application to motion planning using retraction. To justify our claims on practical applicability, we provide experimental results obtained with implementations of the method in two and three dimensions.

1 Introduction

The Voronoi diagram of a set of sites partitions the space into regions such that all points in a region have the same closest site according to some given metric [10]. Voronoi diagrams have proven useful in many problems in the field of computational geometry; we refer to Aurenhammer [3] or Okabe et al. [18] for a survey of the many applications and generalizations of the Voronoi diagram. The great interest in generalized Voronoi diagrams (that is, diagrams of other than point sites) has recently produced many efficient algorithms [5, 7, 14]. However, the low complexity of these algorithms does not automatically make them fit for practical applications because often they are complicated and suffer from robustness problems. To our knowledge, numerically robust algorithms for constructing a topologically consistent approximation of the Voronoi diagram have been proposed only for point sites—either in the plane [20] or in three-space [9]—and line segments in the plane [4]. The latter approach demonstrates that exact implementations are time-consuming because they require calculations to be performed with high precision.

In this paper we propose a different approach that is in our view better suited for particular practical problems. Our interest is in applications for which an approximation of the Voronoi diagram within some predetermined precision is sufficient—motion planning for example. We first describe a general framework to divide the space into a set of primitive cells of fixed size, a subset of which are used to approximate the Voronoi diagram. As primitive cells we

*This research was partially supported by ESPRIT Basic Research Action No. 6546 (project PROMotion) and by the Netherlands Organization for Scientific Research (NWO).

ISSN: 0924–3275

Approximating Generalized Voronoi Diagrams in Any Dimension

Jules Vleugels Mark Overmars

Technical Report UU-CS-1995-14
May 1995

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

Approximating Generalized Voronoi Diagrams in Any Dimension

Jules Vleugels Mark Overmars

UU-CS-1995-14
May 1995