Motion Planning in Environments with Low Obstacle Density *

A. Frank van der Stappen Mark H. Overmars

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

Abstract

We present a simple and efficient paradigm for computing the exact solution of the motion planning problem in environments with a low obstacle density. Such environments frequently occur in practical instances of the motion planning problem. The complexity of the free space for such environments is known to be linear in the number of obstacles. Our paradigm is a new cell decomposition approach to motion planning and exploits properties that follow from the low density of the obstacles in the robot's workspace. These properties allow us to decompose the workspace, subject to some constraints, rather than to decompose the higher-dimensional free configuration space directly. A sequence of uniform steps transforms the workspace decomposition into a free space decomposition of asymptotically the same size. The approach applies to robots with any fixed number of degrees of freedom and turns out to be efficient in many cases: it leads to nearly optimal $O(n \log n)$ algorithms for motion planning in 2D, for motion planning in 3D amidst obstacles of comparable size, and for motion planning on a planar workfloor in 3D. In addition, we obtain algorithms for planning 3D motions among polyhedral obstacles, running in $O(n^2 \log n)$ time, and among arbitrary obstacles, running in time $O(n^3)$. Several other interesting instances of the motion planning seem adequately solvable by the paradigm as well.

1 Introduction

An ultimate goal in the field of robotics is the development of robots that accept high-level descriptions of tasks and execute these tasks without intervention from their environment. A fundamental task for such an autonomous robot would be to move from its current placement to some other specified placement while avoiding collision with the obstacles on its way. The problem of finding such a collision-free path is referred to as the motion planning problem. Even though most of today's operational robots are not fully autonomous, most of them have to deal with certain instances of the motion planning problem during their operation. The methods that are used in practice to tackle these instances have the notable drawback that they may fail to find an existing path (or spend a lot of time and storage to find one). A direction of research in computational geometry, initiated by a series of papers - known as the Piano Movers' series [25, 26, 27, 28, 31] - by Schwartz and Sharir in the early 80s, studies the exact solution of the motion planning problem. Exact methods for solving the

*Research is supported by the Dutch Organization for Scientific Research (N.W.O.) and by the ESPRIT III BRA Project 6546 (PROMotion). A preliminary version of this paper, entitled 'Motion Planning amidst Fat Obstacles', appeared in the Proc. 10th Ann. ACM Symp. on Computational Geometry (1994).

motion planning problem are guaranteed to find a path if one exists, and report failure if no path exists. The disadvantage of exact methods is their high worst-case running time. The high worst-case time bounds prevent exact methods from becoming popular alternatives for the solution of practical instances of the motion planning problem. We show, however, that certain realistic assumptions on the robot and its environment allow for a simple general approach to the solution of exact motion planning problems. The approach leads to several very efficient motion planning algorithms for such instances.

We focus on the following general version of the motion planning problem.

> *Given a robot $\mathcal{B}$ in a workspace* W *with a collection $\mathcal{E}$ of closed connected stationary obstacles, and two placements Z_0 and Z_1, find a motion for the robot from Z_0 to Z_1 during which it avoids collision with the obstacles, or report that no such motion exists.*

The *robot* $\mathcal{B}$ is assumed to be a collection of closed rigid bodies of total constant complexity and to have f degrees of freedom (DOF). The robot moves in a *workspace* W, which usually equals the Euclidean space of dimension two ($\mathbb{R}^2$) or three ($\mathbb{R}^3$). The motion of the robot is constrained by a set $\mathcal{E}$ of n disjoint obstacles. Each *obstacle* $E \in \mathcal{E}$ is a closed connected constant-complexity subset of the workspace W. The obstacles do not change place or shape.

The motion planning problem is commonly modelled and solved in the *configuration space* C, which is the space of parametric representations of robot placements. The dimension of C equals the number of degrees of freedom f of the robot $\mathcal{B}$. A point $Z \in C$ (representing a robot placement) is referred to as a configuration. Although there is a subtle difference between a placement and a configuration, we will use both terms interchangeably. The *free space* FP is the subspace of C consisting of points that represent placements of the robot in which it does not intersect any obstacle in $\mathcal{E}$:

$$\mathrm{FP} = \{Z \in C | \mathcal{B}[Z] \cap (\cup_{E\in\mathcal{E}} E)\},$$

where $\mathcal{B}[Z]$ stands for the set of workspace points covered by $\mathcal{B}$ in configuration Z. The free space can be regarded as the union of certain cells - the free cells - in the arrangement of constraint hypersurfaces. A constraint hypersurface is the set of placements in which a robot feature, i.e., a basic part of the boundary like a vertex, edge, or face, touches an obstacle feature of appropriate dimension. A collision-free *path* or *motion* for a robot $\mathcal{B}$ from an initial placement Z_0 to a final placement Z_1 is a continuous map: $\tau : [0,1] \rightarrow \mathrm{FP}$, with $\tau(0) = Z_0$ and $\tau(1) = Z_1$. Hence, solving the motion planning problem boils down to finding a continuous curve in FP connecting Z_0 and Z_1. The effort that is required to find such a curve clearly depends on the complexity of the free space.

Exact motion planning algorithms process the free space into a query structure that allows for the efficient solution of one or more path-finding queries. Although there essentially exist two different approaches to exact motion planning (cell decomposition and retraction), the time spent in processing the free space and the size of the resulting query structure clearly depend on the complexity of the free space. *Cell decomposition* algorithms (see e.g. [15, 18, 25, 26, 27, 28, 31]) partition the free space into a finite number of simple connected subcells, such that planning a motion between two placements in a single subcell is straightforward and such that uniform crossing rules can be defined for $\mathcal{B}$ crossing from one cell into another. Each cell defines a vertex in the *connectivity graph* CG. Two vertices in CG are connected by an edge if their corresponding subcells share a common boundary allowing direct crossing of the robot. Given the graph CG, the motion planning problem is reduced to a graph problem: determine a sequence of pairwise connected nodes connecting the nodes corresponding to the subcells containing the initial and final placements of $\mathcal{B}$. The imposed simplicity of the subcells facilitates the transformation of the sequence of subcells into an actual collision-free motion for the robot. The desired simplicity of the subcells in the cell decomposition, however, also causes the number of subcells to depend on the complexity of the free space. As a result, the size of the query structure - the connectivity graph CG - and the time to compute it depend on the complexity of FP. *Retraction methods* (see e.g. [8, 17, 20, 21, 32]) aim at capturing the structure and connectivity of the free space in some one-dimensional network of curves in the free space, the *roadmap*. The curves are chosen in such a way that a simple collision-free motion connects every point $Z \in$ FP to some point $\text{Im}(Z)$ on the roadmap, and such that all curves in a single connected component of the free space are connected. Given a roadmap with these properties, the problem of finding a motion between two free placements Z_0 and Z_1 can be reduced to the problem of finding a sequences of roadmap curves connecting the roadmap points $\text{Im}(Z_0)$ and $\text{Im}(Z_1)$. The desired properties of the roadmap, however, also cause the number of curves to depend on the complexity of the free space. As a result, the size of the query structure - the roadmap - and the time to compute it depend on the complexity of FP.

The complexity of the free space is determined by the number of multiple contacts of the robot $\mathcal{B}$. A multiple contact of the robot $\mathcal{B}$ is a placement in which it touches more than one obstacle feature. Besides the collisions of the robot with the obstacles, parts of the robot can also collide with other robot parts. Although these so-called *self-collisions* are often ignored in our considerations, we shall return to them at appropriate moments to demonstrate the validity of the results when self-collisions *are* taken into account. Unfortunately, the number of multiple contacts, and, hence, the complexity of the free space, can be very high. Under our circumstances where the total number of obstacle features is $\Theta(n)$ and the number of features of the f-DOF robot is bounded by a constant, the free space complexity can be $\Omega(n^f)$. As a generic example, consider the robot arm in Figure 1. If the square obstacles are sufficiently small and, within each column, sufficiently close together, then the number of f-fold contacts is easily seen to be $\Omega(n^f)$. As a consequence, the complexity of the free space for the robot arm is $\Omega(n^f)$. Slightly lower worst-case free space complexities have been

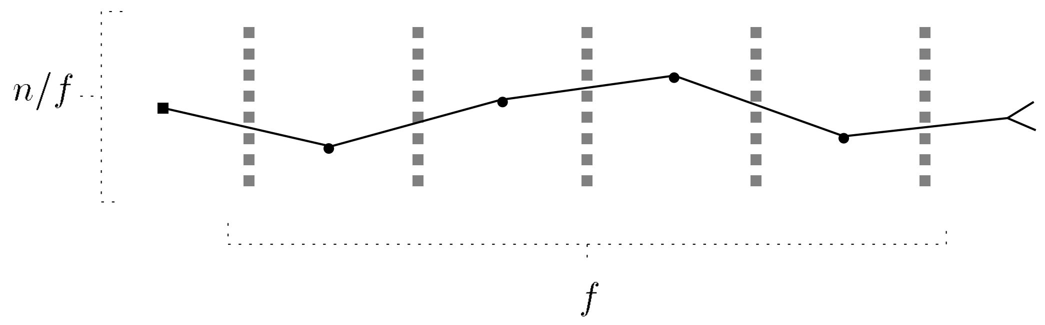

Figure 1: An (f-DOF) robot arm consisting of f links with $\Omega(n^f)$ f-fold contacts, and, hence, with free space complexity $\Omega(n^f)$.

obtained for specific free-flying rigid robots (like convex polyhedra) among certain classes of obstacles (like polyhedra). These bounds generally remain close to one order of magnitude, i.e., a factor n, below the $\Omega(n^f)$ bound (see e.g. [30, 38]). Hence, even in such more specific cases, the theoretical worst-case bounds are high. Fortunately, in many practical situations the complexity of the free space is much smaller, as artificially constructed workspaces with e.g. a very large robot and many small obstacles are not very often encountered in real life. When extreme shapes and sizes of the robot and the obstacles do not occur, high free space complexities tend to be harder to obtain. Consider for example the motion planning environment of Figure 2 where the 6-DOF 'spider' robot and the obstacles have roughly the same sizes. While being in contact with a certain obstacle, the robot is unable to touch more than a constant number of other obstacles. Then, the number of multiple contacts can impossibly exceed $O(n)$. Hence, the free space for this robot has complexity $O(n)$ and thus remains far below the free space complexity obtained with the construction of Figure 1. The

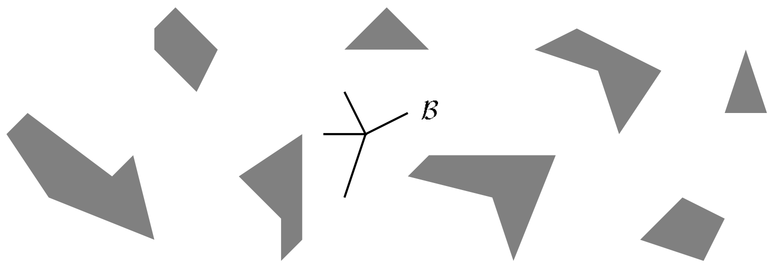

Figure 2: A (6-DOF) robot with few multiple contacts, and, hence, with low free space complexity.

impressive gap between the $\Omega(n^f)$ construction and the $O(n)$ example immediately raises the question what specific properties of the robot and the obstacles lead to low free space complexities. What natural mild assumptions would for example lead to the relative low obstacle density of the above example, in which the robot is unable to touch more than a constant number of obstacles simultaneously?

Van der Stappen, Halperin, and Overmars [33] show that the combinatorial complexity of the free space is linear in the number of obstacles if the robot is not too large compared to the obstacles and if any workspace region of size proportional to the size of the smallest obstacle intersects no more than a constant number of obstacles. We shall refer to the last property as the *low obstacle density* property of the workspace. (Actually, in [33] the linear bound is only proven for the more restricted assumption of fatness of the obstacles. It is though trivial to extend it to sets that satisfy the low obstacle density property.) Circumstances that resemble the low obstacle density have also been studied by Schwartz and Sharir [29] who refer to it as *bounded local complexity* and by Pignon [24] who calls it *sparsity*.

A question that immediately comes to mind when considering the combinatorial result of [33] is whether this reduced complexity opens the way to efficient motion planning algorithms for such realistic environments. The vast majority of motion planning algorithms have no reported sensitivity to the complexity of the free space. A clear exception is the boundary-vertices retraction algorithm by Sifrony and Sharir [32] for a ladder moving in a planar workspace with polygonal obstacles. The algorithm runs in time $O(K \log n)$, where K is the number of pairs of obstacle corners that lie less than the length of the ladder apart. The low obstacle density causes K to be only $O(n)$, whereas it would be $O(n^2)$ for arbitrary workspaces with obstacles. Some algorithms have a hidden sensitivity to the complexity of the free space [36]. For example, the boundary cell decomposition algorithm by Avnaim, Boissonnat, and Faverjon [4], running in time $O(n^3 \log n)$ for a constant complexity polygonal robot amidst arbitrary polygonal obstacles, can be shown to run in $O(n \log n)$ time in the low obstacle density setting. The $O(n^5)$ algorithm by Schwartz and Sharir [25] for planning the motion of a ladder or a polygonal robot amidst polygonal obstacles can be shown to run, unmodified, in time $O(n^2)$ if the obstacle density is low, whereas a minor modification enhances the efficiency to a running time of $O(n \log n)$ (see also [34]). Hence, there exist (planar) motion planning algorithms that do benefit from low free space complexities, even though several other algorithms do not. Algorithms for efficient motion planning in 3D workspaces are scarce: approaches in contact space, like the algorithms mentioned above by Sifrony and Sharir, and by Avnaim, Boissonnat, and Faverjon, were never shown to generalize to higher dimensions. General approaches to motion planning (e.g. by Schwartz and Sharir [26] with running time $O(n^{2^{f+6}})$ and Canny [8] with running time $O(n^f \log n)$) are computationally expensive, even under our beneficial circumstances. These rigorous methods do not take advantage of any accidental structure of the free space (as is present in our case).

In this paper, we present a new paradigm for motion planning in environments with low obstacle density. The idea is that we do not compute a decomposition of the free configuration space but of the (two- or three-dimensional) workspace. Next, we 'lift' this workspace decomposition into the configuration space. We will show that the low obstacle density guarantees that this lifting can be done without increasing the complexity of the decomposition in order of magnitude. The realistic low obstacle density and bounded robot size assumptions also guarantee the existence of small and efficiently computable workspace partitions for various interesting instances of the motion planning problem. We will give linear size partitions for two-dimensional workspaces with arbitrary obstacles and for three-dimensional obstacles of comparable sizes. Both partitions are computable in nearly-optimal $O(n \log n)$ time, so that we obtain $O(n \log n)$ time algorithms for these two instances of the motion planning problem. In addition, we present partitions for three-dimensional workspaces with polyhedral and arbitrary obstacles of quadratic and cubic sizes respectively. The partitions are computable in time $O(n^2 \log n)$ and $O(n^3)$. As a result, we obtain an $O(n^2 \log n)$ algorithm for planning the motion of a robot amidst polyhedral obstacles in 3-space, and an $O(n^3)$ algorithm for planning the motion of a robot amidst arbitrary obstacles in 3-space. A generalization of the ideas, finally, leads to an $O(n \log n)$ algorithm for planning the motion of a (vacuum-cleaner) robot moving on a workfloor among arbitrary obstacles in a three-dimensional workspace. Notice that the running time of the algorithms does not depend on the number f of degrees of freedom of the robot $\mathcal{B}$.

We are aware of only few (related) results on exact motion planning methods with provable efficiency or free space complexity-sensitive behavior for realistic motion planning problems (with low complexity workspaces or free spaces). The running time of Sifrony and Sharir's algorithm [32] depends on the number of pairs of obstacle corners that lie less than the length of the ladder apart. This number gives some idea of how cluttered the obstacles in the workspace are and is closely related to the complexity of the free space. Schwartz and Sharir [29] consider workspaces with obstacles of so-called *bounded local complexity.* Any (imaginary) ball with radius r in such a workspace intersects no more than a constant number of obstacles. The property resembles our notion of low obstacle density. The authors give directions on how to solve the motion planning problem in such workspaces. Pignon [24] structures two-dimensional workspaces with polygonal obstacles and a polygonal robot - using Minkowski differences - to easily detect simple and impossible path-finding queries. Simple queries allow to replace the robot by an outer approximation with fewer degrees of freedom. Non-simple and non-impossible queries require the application of an exact method to the original problem. Alt et al. [2] introduce the tightness of a motion planning problem for a rectangle among polygonal obstacles as a measure for its complexity. The tightness of a problem is closely related to the scaling factor for the rectangular robot to make a solvable problem unsolvable, or an unsolvable problem solvable. The authors present an *approximate* motion planning algorithm for the rectangular robot with a tightness-dependent running time.

This paper is organized as follows. Section 2 formalizes the notion of a low object density space and shows its relation to fatness. It reports some results for low object density spaces that, though interesting in their own right, mainly serve as a tool in the subsequent sections. In Section 3, we exploit the low obstacle density property of the workspace to obtain a paradigm for planning the motion of a robot of bounded size. The running time of algorithms based on the paradigm depends on the time to compute some constrained partition of the workspace. Section 4 discusses a handful of applications of the paradigm: it reports small and efficiently computable partitions of 2D and 3D workspaces with different classes of obstacles. Section 5 concludes the paper.

2 Low obstacle density and its relation to fatness

In many practical situations, the complexity of the free space tends to remain far below the theoretical worst-case complexity bounds. Lower complexities particularly occur when the obstacles in the workspace are not cluttered too much and the robot is not too large compared to the obstacles. A clear but very restrictive example of such an environment is a workspace in which the robot can never touch more than one obstacle at a time. Our aim is to find a weaker and more realistic assumption that still leads to a low free space complexity and efficient motion planning algorithms.

2.1 Low obstacle density

This subsection is devoted to identifying a weak assumption on the workspace and the obstacles so that efficient motion planning is possible. The results are basically reformulations of results previously reported in [33], but we repeat them because they are fundamental to this paper.

As the relative sizes of the robot and the obstacles play a crucial role throughout the paper, we first give convenient measures for the size of an obstacle and a robot. We find the size, or more specifically the radius, of the minimal enclosing hypersphere of an obstacle the most convenient among the many ways to express the obstacle size. The size or radius of the minimal enclosing hypersphere of the robot, however, may vary due to the possibility that the robot may consist of several links. We introduce the reach $\rho_{\mathcal{B}}$ of a robot $\mathcal{B}$ as a means of expressing the size of $\mathcal{B}$. Let $O \in \mathcal{B}$ be the robot's reference point, and assume that the configuration space $C = \mathrm{W} \times D$, where W is the d-dimensional workspace and D is the $(f-d)$-dimensional space of the remaining degrees of freedom.

Definition 2.1 [reach $\rho_{\mathcal{B}}$ of a robot $\mathcal{B}$]
Let Z_{W} be some arbitrary position of the reference point O of the robot $\mathcal{B}$. Then the reach $\rho_{\mathcal{B}}$ of the robot $\mathcal{B}$ is defined as

$$\rho_{\mathcal{B}} = \sup_{Z_D \in D} \max_{p \in \mathcal{B}[(Z_{\mathrm{W}}, Z_D)]} d(p, Z_{\mathrm{W}}).$$

In words, the reach $\rho_{\mathcal{B}}$ of a robot $\mathcal{B}$ is the maximum distance in the workspace that any point in the robot $\mathcal{B}$ can ever have to the reference point, which is also equal to how far the robot can reach, measured from its reference point. Notice the natural similarity of the measures of sizes of the robot and the obstacles: the reach of the robot is the maximum radius that the minimal robot enclosing hypersphere centered at the reference point can ever have (in any placement of $\mathcal{B}$).

The definition in the previous paragraph allow us to impose an explicit bound on the ratio of the sizes of the robot and the obstacles. This bound is one of the two keys to a low free space complexity and to efficient motion planning algorithms. Assuming that the minimal enclosing hypersphere radii of all obstacles in the workspace are at least ρ, the restriction we impose is that the reach $\rho_{\mathcal{B}}$ of the robot $\mathcal{B}$ is bounded by $b \cdot \rho$, for some constant $b \geq 0$. Property 2.2 defines a class of (work)spaces that, in combination with the bound on the relative size of the robot and the obstacles, give rise to a linear complexity free space and allow for efficient motion planning.

Property 2.2 *Let $\mathbb{R}^d$ be a space with a set $\mathcal{E}$ of objects with minimal enclosing hypersphere radii at least ρ. Then $\mathbb{R}^d$ is said to be a* low (object) density space *if any region with minimal enclosing hypersphere radius $c \cdot \rho$, for some constant $c \geq 0$, intersects no more than a constant number of objects $E \in \mathcal{E}$.*

In the specific case that $\mathbb{R}^d$ is the workspace W of a robot, and $\mathcal{E}$ is the set of obstacles in W, we will refer to W as a low obstacle density workspace.

Theorem 2.3 states the linear complexity result. The reader is referred to [33] for a proof.

Theorem 2.3 *The free space for a constant-complexity robot $\mathcal{B}$ with reach $\rho_{\mathcal{B}} \leq b \cdot \rho$ moving in a low obstacle density workspace* W *with n constant-complexity obstacles $E \in \mathcal{E}$ with minimal enclosing hypersphere radii at least ρ, for some constant $b \geq 0$, has complexity $O(n)$.*

In the next subsection, we consider an interesting class of motion planning environments that satisfy Property 2.2. An immediate consequence of the preceding results will be the linear free space complexity for problems in this class.

2.2 Fatness

Fatness has turned out to be an interesting phenomenon in computational geometry. Several papers present surprising combinatorial complexity reductions [2, 10, 16, 19, 33] and efficiency gains for algorithms [1, 6, 14, 22, 23] if the objects under consideration have a certain fatness. Fat objects are compact to some extent, rather than long and thin. Fatness is a realistic assumption, since in many practical instances of geometric problems the considered objects are fat. The aim of studying fatness is to find new fast and simple algorithms or to demonstrate enhanced efficiency of existing algorithms for such practical instances. The achievements of the study of fatness so far include near-linear bounds on the complexity of the union of certain fat figures (e.g. triangles, wedges) in the plane [2, 10, 16, 19], a linear bound on the complexity of the free space for motion planning amidst fat obstacles [33], and efficient algorithms for computing depth orders on certain fat objects [1], binary space partitions for scenes of non-intersecting fat objects in the plane [6], hidden surface removal for fat horizontal triangles [14], and range searching and point location among fat objects [22, 23].

Contrary to many other definitions of fatness in literature [1, 2, 10, 14, 16, 19], the notion introduced in [33], and recaptured below, applies to general objects in arbitrary dimension d. The definition involves a parameter k, supplying a qualitative measure of the fatness of an object: the smaller the value of k, the fatter the object must be.

Definition 2.4 *Let $E \subseteq \mathbb{R}^d$ be an object and let k be a positive constant. The object E is k-fat if for all hyperspheres $S \in U_E$:*

$$k \cdot volume(E \cap S) \geq volume(S),$$

where U_E consists of all hyperspheres centered inside E and not fully containing E.

According to the definition, a k-fat object E must cover at least $1/k$-th of any hypersphere that is centered inside E but not fully contains it. The definition of fatness forbids fat objects to be long and thin, or to have long and thin parts.

Obstacle fatness and low obstacle density are closely related, although this may not seem completely obvious at first sight. An intuitive explanation lies in the observation that it is impossible to have a very large number of fat obstacles of a certain minimum size intersecting a small region. A more formal proof follows from [33]. Here, we confine ourselves to reporting the result.

Theorem 2.5 *A space $\mathbb{R}^d$ with non-intersecting k-fat objects is a low object density space.*

2.3 Object wrappings

This subsection studies the complexity implications of expanding the objects in a scene satisfying Property 2.2 for the arrangement of object boundaries. Clearly, the arrangement of the boundaries of n non-intersecting constant-complexity objects has $O(n)$ complexity. Let us see what happens if the objects are expanded. While expanding the objects, each of the boundaries will eventually start intersecting other boundaries. Intuitively, the first boundaries that are to be intersected belong to neighboring objects. As the density of other objects in the vicinity of each object is low, a considerable expansion of the objects is, again intuitively, necessary to create more than a few intersections of object expansions, and, hence, to asymptotically increase the complexity of the arrangement of boundaries. Below, these informal ideas are made specific by giving accurate bounds on the (allowable) expansion of the objects such that the combinatorial complexity of the arrangement of the boundaries of these (intersecting) expansions remains $O(n)$. The so-called ϵ-wrappings that are introduced provide a convenient means of expressing the expansion of an object.

Sufficiently tight wrappings play a crucial role in providing the justification that the paradigm for motion planning in low obstacle density workspaces presented in Section 3 indeed works. Besides that, the wrappings also help in finding efficient instances of the paradigm, for specific classes of motion planning problems. Moreover, the theorem on wrappings that we prove below is interesting in its own right, as it implies nice complexity bounds for certain types of arrangements and for the boundary of the union of specific families of shapes.

Definition 2.6 [**ϵ-wrapping**]
Let $E \subseteq \mathbb{R}^d$ and let $\epsilon \in \mathbb{R}^+$. Any object Δ satisfying $E \subseteq \Delta \subseteq \{ p \in \mathbb{R}^d \mid d(p,E) \leq \epsilon \}$ is an ϵ-wrapping of E.

An ϵ-wrapping of an object E is an enclosing shape of E, with the property that the distance from the wrapping to E never exceeds ϵ.

Theorem 2.7 states the circumstances that lead to a linear complexity arrangement of expanded object boundaries. An obvious way to express a bound on the expansion of an object E is to state that the expanded object is some ϵ-wrapping of the object E itself, for some bounded positive ϵ.

Theorem 2.7 *Let $\mathbb{R}^d$ be a low object density space with a set $\mathcal{E}$ of n non-intersecting objects with minimal enclosing hypersphere radii at least ρ. Let $c \geq 0$ be some constant and assume that a constant-complexity $(c \cdot \rho)$-wrapping Δ_E is given for every object $E \in \mathcal{E}$. Then:*

(a) *the complexity of the arrangement $\mathcal{A}(\Delta)$ of all wrapping boundaries $\partial\Delta_E$ is $O(n)$,*

(b) *every point $p \in \mathbb{R}^d$ lies inside at most $O(1)$ wrappings Δ_E.*

Proof: Let us assume that the objects in $\mathcal{E}$ are ordered by increasing minimal enclosing hypersphere radius: $E_1, \ldots, E_n$ and $\rho \leq \rho_1 \leq \ldots \leq \rho_n$, where ρ_i is the minimal enclosing hypersphere radius of E_i. Let Δ_i $(1 \leq i \leq n)$ be the $(c \cdot \rho)$-wrapping corresponding to E_i. Finally, let S_i be the hypersphere with radius $(2c+1) \cdot \rho_i$ concentric E_i's minimal enclosing hypersphere.

We intend to count for each object E_i the subspaces of dimensions 0 through $d-1$ that are defined by the intersection of $\partial\Delta_i$ and wrapping boundaries $\partial\Delta_j$ with $j > i$. A $(c \cdot \rho)$-wrapping boundary $\partial\Delta_i$ can only be intersected by $(c \cdot \rho)$-wrapping boundaries $\partial\Delta_j$ $(j > i)$ if the distance from E_j to E_i does not exceed $2c \cdot \rho \leq 2c \cdot \rho_i$. The object E_j must then clearly intersect the hypersphere S_i (with diameter $(4c+2) \cdot \rho_i$. Property 2.2 yields that there can only be a constant number of such E_j's, so there is at most a constant number of wrapping boundaries $\partial\Delta_j$ $(j > i)$ that intersect $\partial\Delta_i$. By the additional assumption that all wrappings have constant complexity, there is only a constant number of constant-complexity subspaces of dimension between 0 and $d-1$ defined by the intersection of $\partial\Delta_i$ and wrapping boundaries $\partial\Delta_j$ $(j > i)$. Adding the contributions of all wrappings amounts to a total of $O(n)$ subspaces of dimensions 0 to $d-1$ in the arrangement $\mathcal{A}(\Delta)$. The linear bounds on the number of these subspaces imply the same bound of $O(n)$ on the number of d-faces in $\mathcal{A}(\Delta)$, making the total combinatorial complexity of the arrangement $O(n)$.

The (b)-part follows immediately from the proof of the (a)-part. Let E_i be the smallest object for which the point $p \in \mathbb{R}^d$ lies inside the wrapping Δ_i. Since there is only a constant number of wrappings of larger objects intersecting E_i's wrapping, the point p can be in no more than a constant number of additional wrappings. $\square$

Besides applications in motion planning, Theorem 2.7 has interesting implications for complexities of union boundaries of certain geometric figures. The relation between the complexity of an arrangement of wrapping boundaries and the complexity of the boundary of the union of the wrappings becomes clear if one realizes that the faces of the union boundary form a subset of the faces of the arrangement of wrapping boundaries. So, under the circumstances sketched in Theorem 2.7, the boundary of the union of all wrappings Δ has complexity $O(n)$.

The result of the theorem is, for example, applicable to the molecule model in the paper by Halperin and Overmars [12]. The atoms that constitute a molecule are assumed to satisfy the hard sphere model. The hard sphere model describes atoms by spheres and forbids any sphere center to penetrate another sphere too far. This property makes it possible to regard the atoms as wrappings of certain non-intersecting smaller spheres, which are only a bounded amount smaller than the original atoms. The construction provides an alternative proof for the linear (in the number of atoms) descriptional complexity of the molecule surface.

3 A paradigm for motion planning in environments with low obstacle density

The ultimate aim of this paper is to determine a general approach to planning the motion of a not too large, constant-complexity robot moving in a workspace with a low density of constant-complexity obstacles. It has been noted that the existing planar motion planning algorithms are not easily extendible towards other - in particular spatial - problems. Moreover, the existing general approaches to motion planning (like those by Schwartz and Sharir [26] and Canny [8]) are computationally expensive, even for problems from the special class that we consider here.

Motion planning problems in Euclidean workspaces of dimension three normally imply at least three-dimensional configuration spaces. A configuration space contains constraint hypersurfaces of the form $f_{\phi,\Phi}$, consisting of placements of the robot $\mathcal{B}$ in which a robot feature ϕ is in contact with an obstacle feature Φ. We shall denote the fact that ξ is a feature of some object or object set X by $\xi \in_f X$. The arrangement of all (constant-complexity) constraint hypersurfaces $f_{\phi,\Phi}$ ($\phi \in_f \mathcal{B}, \Phi \in_f \mathcal{E}$) divides the higher-dimensional configuration space into free cells and forbidden cells. Even in the case of low density motion planning, the complexity of a single free cell can be $O(n)$, which illustrates that some additional processing is necessary to facilitate efficient motion planning. Naturally, the structure of a higher-dimensional arrangement like the arrangement of constraint hypersurfaces is difficult to understand, let alone to subdivide the free arrangement cells into simple subcells or to catch their structure in some one-dimensional roadmap. At this point, however, the low obstacle density comes to our help to provide us with a very useful property of an f-dimensional configuration space C of the form $C = \mathrm{W} \times D$, where W is the d-dimensional workspace and D is some $(f-d)$-dimensional (rest-)space. (Free-flying rigid robots, for example, fit well in this framework. For a free-flying rigid robot in $\mathrm{W} = \mathbb{R}^3$, D is the space defined by the three rotational degrees of freedom of the robot.) The low obstacle density can be shown to result in a very interesting property of configuration space, namely that for each point $p \in \mathrm{W}$:

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (p \times D) \neq \emptyset\}| = O(1).$$

In words, the $(f-d)$-dimensional subspace $p \times D$, obtained by lifting the workspace point p into configuration space, is intersected by only a constant number of constraint hypersurfaces. An immediate consequence of this result is that the hypersurfaces define a constant-complexity arrangement in each cross-section $p \times D$ of the configuration space C.

At a more abstract level, low obstacle density motion planning problems for free-flying robots can be regarded as a subclass of the larger class of motion planning problems with configuration spaces $C = B \times D$ that satisfy for each point $p \in B$:

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (p \times D) \neq \emptyset\}| = O(1).$$

A configuration space C that satisfies this constraint will be said to be *cylindrifiable.* Furthermore, we call the subspace B of C a *base space.* Hence, low obstacle density motion planning problems for free-flying robots have cylindrifiable configuration spaces in which the workspace constitutes a valid base space. As a result of the cylindrifiability of C, it is possible to partition the subspace B into closed regions R (or C into cylinders $R \times D$) such that

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D) \neq \emptyset\}| = O(1).$$

The partition of B that leads to the cylinders will be called the *base partition* (corresponding to the cylindrical decomposition). Figure 3 illustrates the terminology introduced in this paragraph.

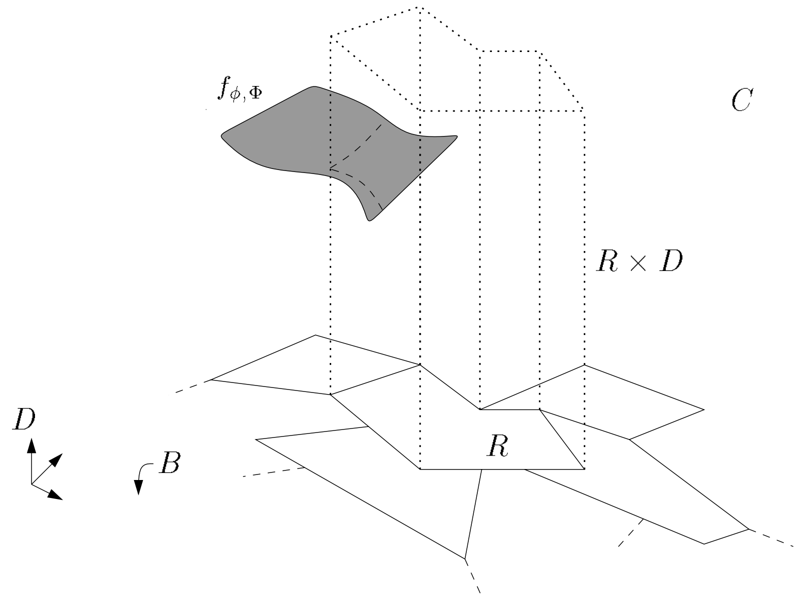

Figure 3: A three-dimensional example of a cylindrifiable configuration space C with a two-dimensional base space B (and, hence, a one-dimensional rest-space D), and a fragment of the base partition in B. The constraints on the base partition guarantee that the cylinder $R \times D$ is intersected by only a constant number of constraint surfaces $f_{\phi,\Phi}$.

Let us now consider the configuration space cylinder $R \times D$ corresponding to a region R in a base partition in B. By the definition of the base partition, the cylinder $R \times D$ is intersected by $O(1)$ constraint hypersurfaces. These hypersurfaces subdivide the cylinder $R \times D$ into a constant number of cells, due to their constant complexity. If we furthermore assume that the cylinders themselves have constant descriptional complexity (achievable by establishing that the regions R have constant complexity) then each of the $O(1)$ (free or forbidden) cells in $R \times D$ has constant complexity as well. In conclusion, the constraint hypersurfaces and the cylinder boundaries divide the free space into constant-complexity, and hence simple, subcells.

The preceding arguments suggest a two-step approach for computing a cell decomposition for a motion planning problem with a cylindrifiable configuration space: first, find a base partition in some appropriate base space B of C, and then transform the partition into a cell decomposition of the free space $\mathrm{FP} \subseteq C$, by computing a decomposition of the free part of every cylinder. We shall see that the resulting decomposition consists of subcells that allow for simple motion planning within their interiors, and that the rules for crossing from one cell into another are simple.

In Subsection 3.1, it is shown how the latter part of the two-step approach outlined above transforms a base partition into a cell decomposition of comparable size in time proportional to the size of the base partition. Noting this, the problem of finding a (small) cell decomposition of the free space $\mathrm{FP} \subseteq C$ reduces to the problem of finding a (small) base partition in an appropriate base space $B \subseteq C$. Section 3.2 exploits specific properties of the constraint hypersurfaces that follow from the shapes and relative positions of the obstacles to simplify the constraints on the partition of the base space $B = \mathrm{W}$ for motion planning problems involving free-flying robots. The new and simpler constraints combined with the transformation steps result in a tailored paradigm for motion planning for robots in environments with low obstacle density. In Section 4, this paradigm is shown to lead to efficient algorithms for planning motions for free-flying robots amidst several types of obstacles in different workspaces.

3.1 Transforming a base partition into a cell decomposition

We consider a motion planning problem for a constant-complexity robot $\mathcal{B}$ amidst constant-complexity obstacles $E \in \mathcal{E}$. Pairs consisting of a feature $\phi \in_f \mathcal{B}$ and a feature $\Phi \in_f \mathcal{E}$ of matching dimension define constraint hypersurfaces $f_{\phi,\Phi}$ in the cylindrifiable configuration space $C = B \times D$. Furthermore, we assume that we are given a graph (V_B, E_B), where V_B is a set of constant-complexity closed regions R that partition B and individually satisfy

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D) \neq \emptyset\}| = O(1),$$

and E_B contains the adjacencies of V_B's regions: $E_B = \{(R, R') \in V_B \times V_B | \partial R \cap \partial R' \neq \emptyset\}$. Note that in this definition, every region is adjacent to itself. This is crucial to the correctness of the algorithm below.

The transformation algorithm transforms the graph (V_B, E_B) into a connectivity graph $\mathrm{CG} = (V_C, E_C)$, consisting of a set V_C of constant-complexity subcells that collectively partition the set of free placements FP, and a set $E_C = \{(A, A') \in V_C \times V_C | \partial A \cap \partial A' \neq \emptyset\}$ of subcell adjacencies. The sizes of the sets V_C and E_C are of the same order of magnitude as the sizes of V_B and E_B respectively: $|V_C| = O(|V_B|)$, $|E_C| = O(|E_B|)$. Note that the graph (V_C, E_C) supports simple path-finding between two placements in subcells $A \in V_C$ and $A' \in V_C$: the constant complexity of the individual subcells guarantees easy path-finding within a subcell, and the constant complexity of the shared boundary of two adjacent subcells - following from the constant complexity of the involved subcells - caters for simple boundary crossing rules. The transformation steps are, contrary to the computation of the base partition, independent of the actual motion planning problem under consideration.

```
TRANSFORM : $(V_B, E_B) \longrightarrow (V_C, E_C)$

$V_C := \emptyset$;
$E_C := \emptyset$;
for all $R \in V_B$ do
    1. compute the arrangement $\mathcal{A}$ of surfaces $f_{\phi,\Phi}$ intersecting $R \times D$;
    2. use $\mathcal{A}$ to compute a decomposition of $\mathrm{FP} \cap (R \times D)$ into
```

```
            a constant number of constant-complexity subcells $A$;
        3. $Desc(R) := \emptyset$;
        4. **for all** subcells $A$ of $\mathrm{FP} \cap (R \times D)$ **do**
            4.1. $V_C := V_C \cup \{A\}$;
            4.2. $Desc(R) := Desc(R) \cup \{A\}$;
    **for all** $(R_1, R_2) \in E_B$ **do**
        **for all** $A_1 \in Desc(R_1) \wedge A_2 \in Desc(R_2)$ **do**
            **if** $\partial A_1 \cap \partial A_2 \neq \emptyset$ **then** $E_C := E_C \cup \{(A_1, A_2)\}$.
```

Figure 4 gives a pictorial explanation of the transformation.

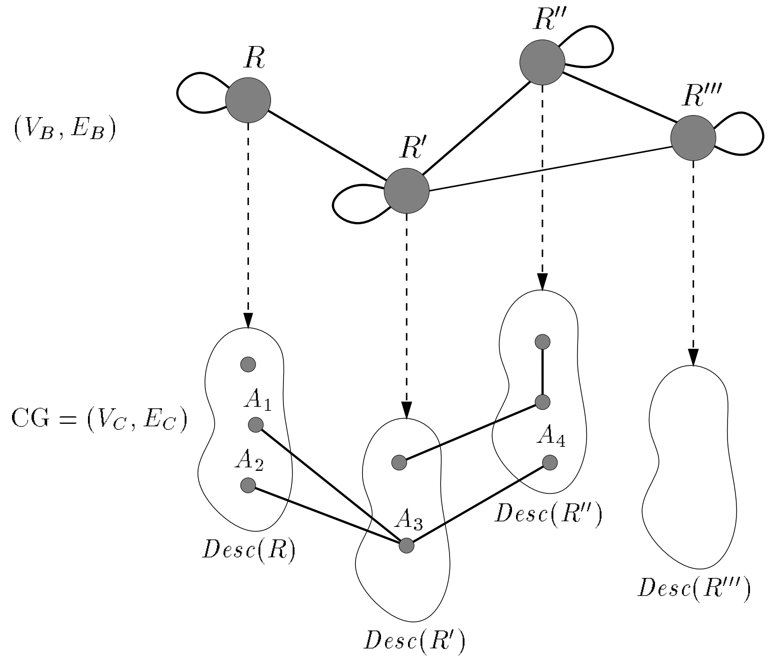

Figure 4: The relation between the base partition graph (V_B, E_B) in the subspace B of C at the top, and the connectivity graph $\mathrm{CG} = (V_C, E_C)$ in the configuration space at the bottom. Each node/region $R \in V_B$ defines at most $O(1)$ nodes $A \in V_C$, collected in a set $Desc(R)$. Two nodes A and A' in V_C can only be connected if the corresponding nodes R and R' in V_B are connected, so, for example, A_1 may be connected to all nodes in $Desc(R')$, but A_1 and A_4 can never be connected.

We review the different steps of the transformation in more detail to verify their validity and to determine the efficiency. Recall that the definition of the set V_B and the constant complexity of the regions $R \in V_B$ imply the constant complexity of all subcells $A \in V_C$.

The first **for**-loop computes a decomposition of $\mathrm{FP} \cap (R \times D)$ into $O(1)$ (constant-complexity) subcells and gathers these in a set V_C. One possible way to perform this computation in constant time is by applying (in step 2) the rigorous techniques by Schwartz and Sharir [26] to the constant number of constraint hypersurfaces intersecting the cylinder $R \times D$. The outcome is a subdivision of the arrangement cells into constant-complexity subcells. Restriction of these subcells to $R \times D$ and subsequently filtering out the forbidden ones results in an appropriate cell decomposition of $\mathrm{FP} \cap (R \times D)$. Each of the four steps in the loop is easily verified to run in constant time, provided that the constraint surfaces $f_{\phi,\Phi}$ intersecting $R \times D$ can be determined in constant time. In future applications of the transformation algorithm, we shall take care that this precondition is fulfilled. If the requirement is indeed settled, the entire loop runs in time $O(|V_B|)$. Upon termination of the first loop, each set $Desc(R)$ stores all nodes in V_C that correspond to free subcells in $R \times D$. Note that each set $Desc(R)$ has constant cardinality.

Two free subcells A_1 and A_2 are adjacent if they share a common boundary (which allows for collision-free crossing from one subcell into the other). Such subcells A_1 and A_2 can only be adjacent if their containing cylinders $R_1 \times D \supseteq A_1$ and $R_2 \times D \supseteq A_2$ are adjacent in C and, hence, R_1 and R_2 are adjacent in B. An adjacency (R_1, R_2) gives rise to only a constant number of adjacencies of nodes A_1 and A_2 in $Desc(R_1)$ and $Desc(R_2)$ respectively due to the constant cardinality of $Desc(R_1)$ and $Desc(R_2)$. Two free subcells A_1 and A_2 in adjacent cylinders are adjacent if they share a common boundary. Such a common boundary has constant complexity since both involved free subcells have constant complexity. The nested **for**-loop in the second **for**-loop takes constant time by the above considerations, implying a running time of $O(|E_B|)$ for the latter loop. If we combine the time-bounds of the two steps in the transformation algorithm, then we find that the running time depends solely on the size of the base partition in a lower-dimensional subspace of the configuration space.

Lemma 3.1 *The algorithm* Transform *transforms the graph* (V_B, E_B) *corresponding to a base partition of the base space* B *into the connectivity graph* (V_C, E_C) *of a cell decomposition of the free space* $\mathrm{FP} \subseteq C = B \times D$ *in time* $O(|V_B| + |E_B|)$.

Once we have computed the connectivity graph (V_C, E_C), the problem of solving a motion planning query 'find a free path from a placement $Z_1 = (Z_{1B}, Z_{1D})$ to another placement $Z_2 = (Z_{2B}, Z_{2D})$' basically reduces to a point location query with Z_{1B} and Z_{2B} in V_B to find $R_1 \ni Z_{1B}$ and $R_2 \ni Z_{2B}$. So, we need a structure for point location in the base space rather than in the full configuration space C. After that it takes $O(1)$ to find $A_1 \ni Z_1$ using $Desc(R_1)$ and $A_2 \ni Z_2$ using $Desc(R_2)$, followed by a search in the graph (V_C, E_C) for a sequence of subcells connecting A_1 to A_2. The constant complexities of the subcells and of the common boundaries of pairs of adjacent subcells facilitate the transformation of the subcell sequence into an actual free path for $\mathcal{B}$.

3.2 A tailored paradigm for free-flying robots

We now direct our attention to a subset of the class of motion planning problems with cylindrifiable configuration spaces, namely the class of problems involving a not too large constant-complexity robot $\mathcal{B}$ with f degrees of freedom moving in a workspace with constant-complexity obstacles $E \in \mathcal{E}$ that satisfies Property 2.2, where f is a constant. The restriction on the size of the robot is expressed by a bound on its reach: $\rho_{\mathcal{B}} \leq b \cdot \rho$, where b is some positive constant and ρ is a lower bound on the minimal enclosing hypersphere radii of the n obstacles in $\mathcal{E}$. For the moment, we assume that the robot $\mathcal{B}$ does not self-collide, that is, no part of $\mathcal{B}$ can collide with any other part of $\mathcal{B}$ during motion. Let $O \in \mathcal{B}$ be the reference point of the robot. The tailored paradigm presented below suits robots with configuration spaces

$$C = \mathrm{W} \times D,$$

so that the position of the robot's reference point in the robot's workspace is part of the specification of its placement. A placement Z of the robot can thus be written as $Z = (Z_{\mathrm{W}}, Z_D)$, where $Z_{\mathrm{W}} \in \mathrm{W} = \mathbb{R}^d$ and $Z_D \in D$. Free-flying robots fit naturally in this framework. Examples for the rest-space D are $D = [0, 2\pi)$ for a free-flying rigid robot in the plane, and $D = [0, 2\pi)^2 \times [0, \pi]$ for a similar robot in three-dimensional space.

In fact, the ideas outlined below are rather easily seen to apply to the larger class of problems with configuration spaces of the form $C = B \times D$, where B is some projective subspace of the workspace W, given that $p_B \in B$ suffices to fix the position of the robot's reference point in the workspace. At first sight, this may seem like an impractical generalization. Imagine, however, a vacuum-cleaning robot which moves in a three-dimensional workspace although its motion is restricted to a plane (the floor). Here, a point $p_B \in B = \mathbb{R}^2$ is sufficient to describe the position of the vacuum cleaner's reference point. The problem of finding a cell decomposition of the free space is in this specific case reduced to a problem in a two-dimensional subspace of the workspace. (See Subsection 4.5 for details on this type of problems.)

Let us return to motion planning problems with $C = \mathrm{W} \times D$. The robot with its reference point fixed at p can only touch obstacles within a distance $\rho_{\mathcal{B}}$ from the point p. Such obstacles clearly intersect the hypersphere with radius $\rho_{\mathcal{B}}$ centered at p. Property 2.2 implies that the number of obstacles with minimal enclosing hypersphere radii at least ρ intersecting any region with diameter $2\rho_{\mathcal{B}} \leq 2b \cdot \rho$ is bounded by a constant. As all obstacles in $\mathcal{E}$ have minimal enclosing hypersphere radii at least ρ, the robot $\mathcal{B}$ can touch no more than $O(1)$ obstacles while its reference point remains fixed at p. This fact leads to the following lemma, which validates the choice of W as a base space for the cylindrical cell decomposition.

Lemma 3.2 *For all $p \in \mathrm{W}$:*

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (p \times D) \neq \emptyset\}| = O(1).$$

Proof: The subspace $p \times D$ of the configuration space is intersected by constraint hypersurfaces $f_{\phi,\Phi}$. A point in $f_{\phi,\Phi} \cap (p \times D)$ corresponds to a placement of the robot $\mathcal{B}$ in which its reference point is positioned at p and its feature ϕ touches an obstacle feature Φ. This feature Φ must necessarily belong to one of the $O(1)$ obstacles that can be touched by $\mathcal{B}$ while its reference point is fixed at p. Combined with the constant complexity of $\mathcal{B}$ itself, this implies that there exist only a constant number of pairs (ϕ, Φ) for which $f_{\phi,\Phi}$ intersects $p \times D$. $\square$

In the sequel we define a partition of the workspace that is subject to constraints that are formulated exclusively in the workspace. The partition subsequently turns out to be a valid base partition for a cylindrical decomposition of the configuration space.

We define the notion of grown obstacles to formalize the observation that the robot $\mathcal{B}$ is unable to touch an obstacle E if the distance from the location of $\mathcal{B}$'s reference point to the obstacle E exceeds $\rho_{\mathcal{B}}$.

Definition 3.3 [grown obstacle $G(E, \rho)$]
Let E be an obstacle in $\mathbb{R}^d$ and let $\rho \in \mathbb{R}^+$. The ρ-grown obstacle E is defined as:

$$G(E, \rho) = \{\, p \in \mathbb{R}^d \,|\, d(p, E) \leq \rho \,\}.$$

Note that, as an alternative definition, the ρ-grown obstacle $G(E, \rho)$ equals the Minkowski sum of E and the hypersphere $S_{O,\rho}$ with radius ρ centered at the origin, so $G(E, \rho) = E \oplus S_{O,\rho}$, where $\oplus$ denotes the Minkowski sum operator. Clearly, the robot's reference point must lie inside $G(E, \rho_{\mathcal{B}})$ in order for the robot $\mathcal{B}$ to be in contact with E; if the reference point lies outside $G(E, \rho_{\mathcal{B}})$ there is no danger for $\mathcal{B}$ of colliding with E. A formalization of these observations leads to a very interesting property on the 'location' of a constraint hypersurface in configuration space.

Lemma 3.4 *Let $\phi \in_f \mathcal{B}$ and $\Phi \in_f E$. Then:*

$$f_{\phi,\Phi} \subseteq G(E, \rho_{\mathcal{B}}) \times D.$$

Proof: The arguments of the proof are given in the workspace. Let $p = (p_{\mathrm{W}}, p_D) \in f_{\phi,\Phi}$, such that $p_{\mathrm{W}} \in \mathrm{W}$ and $p_D \in D$. We must prove that $p = (p_{\mathrm{W}}, p_D) \in G(E, \rho_{\mathcal{B}}) \times D$, which may be reduced to proving that $p_{\mathrm{W}} \in G(E, \rho_{\mathcal{B}})$, since $p_D \in D$ is trivially true. This means that it should be proven that $\mathcal{B}$'s reference point must be placed inside $G(E, \rho_{\mathcal{B}})$ when $\mathcal{B}$'s feature ϕ touches Φ.

Assume, for a contradiction, that $p_{\mathrm{W}} \notin G(E, \rho_{\mathcal{B}})$. Then, by the definition of a grown obstacle, the distance from p_{W} to E exceeds $\rho_{\mathcal{B}}$. But then, it is impossible for $\mathcal{B}$ to reach (and touch) the obstacle E, by the definition of the reach of a robot. In other words, no feature $\phi' \in_f \mathcal{B}$ can touch a feature $\Phi' \in_f E$. So, the point $p = (p_{\mathrm{W}}, p_D)$ with $p_{\mathrm{W}} \notin G(E, \rho_{\mathcal{B}})$ cannot lie on $f_{\phi,\Phi}$ contradicting the assumption of the lemma. □

The lemma supplies some kind of a simple outer approximation of the location of a constraint hypersurface in configuration space. If a workspace region R does not intersect a grown obstacle $G(E, \rho_{\mathcal{B}})$ then certainly none of the constraint hypersurfaces $f_{\phi,\Phi}$ with $\Phi \in_f E$ intersects the configuration space cylinder $R \times D$. If on the other hand, R intersects $G(E, \rho_{\mathcal{B}})$, then one or more constraint hypersurfaces $f_{\phi,\Phi}$ with $\Phi \in_f$ may (but not necessarily must) intersect $R \times D$. As a result, the configuration space cylinder $R \times D$ corresponding to a region R that is intersected by $O(1)$ grown obstacles is itself intersected by at most $O(1)$ constraint hypersurfaces. The following definition of the coverage of a workspace region facilitates a compact statement of this result.

Definition 3.5 [coverage $Cov(R)$]
Let $R \subseteq \mathrm{W} = \mathbb{R}^d$.

$$Cov(R) = \{\, E \in \mathcal{E} \mid R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset \,\}.$$

Hence, $Cov(R)$ is the set of obstacles E whose corresponding grown obstacles $G(E, \rho_{\mathcal{B}})$ intersect R. We use the definition to formulate and prove the relation between the grown obstacles in the workspace and the constraint hypersurfaces in the configuration space.

Lemma 3.6 *Let $R \subseteq \mathrm{W} = \mathbb{R}^d$ be such that $|Cov(R)| = O(1)$. Then*

$$|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D) \neq \emptyset\}| = O(1).$$

Proof: Take a constraint hypersurface $f_{\phi,\Phi} \cap (R \times D) \neq \emptyset$. Now let E be such that $\Phi \in_f E$. By Lemma 3.4, $f_{\phi,\Phi} \subseteq G(E, \rho_{\mathcal{B}}) \times D$. Hence, necessarily $(R \times D) \cap (G(E, \rho_{\mathcal{B}}) \times D) \neq \emptyset$ and thus $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$. By the definition of $Cov(R)$ and the assumption $|Cov(R)| = O(1)$, it follows that there are only $O(1)$ obstacles E such that $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$. Due to the constant complexity of these obstacles and the robot, there is only a constant number of hypersurfaces $f_{\phi,\Phi}$ with $f_{\phi,\Phi} \cap (R \times D) \neq \emptyset$. □

The lemma states that any region R with $|Cov(R)| = O(1)$ is guaranteed to satisfy the constraint on the regions of the base partition requiring that the corresponding cylinder is intersected by $O(1)$ constraint hypersurfaces. As a consequence, a decomposition of the workspace

W into constant-complexity regions R with $|Cov(R)| = O(1)$ is a valid base partition of the base space $B = \mathrm{W}$. We shall refer to workspace partitions of this kind as cc-partitions (constant-size coverage, constant-complexity).

Definition 3.7 [cc-partition]
A cc-partition V of a workspace W *with obstacles $\mathcal{E}$ is a partition of* W *into constant-complexity regions R satisfying $|Cov(R)| = O(1)$.*

The constant-size coverage constraint $|Cov(R)| = O(1)$ replaces the constraint $|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (R \times D) \neq \emptyset\}| = O(1)$; the new constraint is simpler because it is truly a constraint in the workspace. The result in Lemma 3.6 and the definition of cc-partitions, however, would be completely useless if a partition of W into regions R with $|Cov(R)| = O(1)$ does not exist. Note that the existence of such a partition solely depends on the absence of points $p \in \mathrm{W}$ that are contained in $\omega(1)$ grown obstacles. Fortunately, such points do indeed not exist by Lemma 3.8. The lemma follows immediately from Theorem 2.7, noting that each grown obstacle $G(E, \rho_{\mathcal{B}})$ is a constant-complexity $\rho_{\mathcal{B}}$-wrapping and, by $\rho_{\mathcal{B}} \leq b \cdot \rho$, also a constant-complexity $(b \cdot \rho)$-wrapping of the obstacle E itself.

Lemma 3.8 *Let $\mathrm{W} = \mathbb{R}^d$ be a low obstacle density workspace with a set $\mathcal{E}$ of n non-intersecting obstacles with minimal enclosing hypersphere radii at least ρ. Furthermore, let $\rho_{\mathcal{B}} \leq b \cdot \rho$, for some positive constant b. Then*

- **(a)** *the complexity of the arrangement $\mathcal{A}(G)$ of all grown obstacle boundaries $\partial G(E, \rho_{\mathcal{B}})$ $(E \in \mathcal{E})$ is $O(n)$,*
- **(b)** *every point $p \in \mathrm{W} = \mathbb{R}^d$ lies in at most $O(1)$ grown obstacles $G(E, \rho_{\mathcal{B}})$ $(E \in \mathcal{E})$.*

Lemma 3.8(b) shows that it is possible to partition the low obstacle density workspace W into regions with constant-size coverage. Notice that the arrangement $\mathcal{A}(G)$ even partitions $\mathrm{W} = \mathbb{R}^d$ into $O(n)$ regions R with $|Cov(R)| = O(1)$, as each d-cell of the arrangement is a subset of the intersection of $O(1)$ grown obstacles (by Lemma 3.8(b)). Unfortunately, the partition does not suit our purposes, because the d-cells themselves may have more than constant complexity. Hence, it is not a cc-partition. It can though be further refined into one.

In summary, we have found (Lemma 3.8) that a cc-partition of a low obstacle density workspace always exists. The cc-partition in the workspace corresponds, by Lemma 3.6, to a decomposition of the configuration space into constant-complexity cylinders that are intersected by no more than a constant number of constraint hypersurfaces. As a result, the cc-partition is a valid partition of the base space W allowing for application of the transformation algorithm from Subsection 3.1.

The compact algorithm LDMot given below combines the search for a small cc-partition with its transformation into a cell decomposition of the free space. Besides the cc-partition regions, gathered in a set V_{W}, the first step is to report the adjacencies of the cc-partition regions in a set E_{W}, and the function $Cov : V_{\mathrm{W}} \rightarrow \mathcal{P}(\mathcal{E})$ mapping each region $R \in V_{\mathrm{W}}$ onto the (constant-cardinality) set of obstacles $E \in \mathcal{E}$ with $G(E, \rho_{\mathcal{B}}) \cap R \neq \emptyset$. (Occasionally, the pair $(V_{\mathrm{W}}, E_{\mathrm{W}})$ will be referred to as a cc-partition graph.) We denote the time required to compute the triple $(V_{\mathrm{W}}, E_{\mathrm{W}}, Cov)$ by $T(n)$, where the argument n represents the number of obstacles in $\mathcal{E}$.

```
Algorithm LDMot

Find a cc-partition graph $(V_W, E_W)$ and compute $Cov : V_W \to \mathcal{P}(\mathcal{E})$;
$(V_C, E_C) :=$ Transform($(V_W, E_W)$)
```

The $|V_W|$ precomputed sets $Cov(R)$ facilitate the constant-time computation of the constraint hypersurface arrangement $\mathcal{A}$ in step 1 of the first **for**-loop of the transformation algorithm (Transform). To verify this statement, we refine that step to

```
1.1. $F := \emptyset$;
1.2. for all $\phi \in_f \mathcal{B} \wedge \Phi \in_f Cov(R)$ do
        1.2.1. compute $f_{\phi,\Phi}$;
        1.2.2. $F := F \cup \{f_{\phi,\Phi}\}$;
1.3. compute the arrangement $\mathcal{A}$ of all $f \in F$;
```

A closer look at the refinement learns that $\mathcal{A}$ is now the arrangement of all constraint hypersurfaces in a set $F = \{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f Cov(R)\}$, which is in fact a superset of the set of hypersurfaces $f_{\phi,\Phi}$ satisfying $f_{\phi,\Phi} \cap (R \times D) \neq \emptyset$. Fortunately, the easily computable set F contains only a constant number of hypersurfaces, due to the constant cardinality of $Cov(R)$ and the constant complexity of $\mathcal{B}$ and the individual obstacles E. Crucial to the validity of the approach of computing some larger arrangement is the simple observation that $\mathcal{A} \cap (R \times D)$, i.e., the restriction of the arrangement $\mathcal{A}$ to the cylinder $R \times D$, is equivalent to the restriction to $R \times D$ of the arrangement of hypersurfaces $f_{\phi,\Phi}$ with $f_{\phi,\Phi} \cap (R \times D) \neq \emptyset$. The techniques by Schwartz and Sharir from [26] may be useful to compute a decomposition of the free part $\mathrm{FP} \cap (R \times D)$ of a cylinder $R \times D$.

The refinement of step 1 of the first **for**-loop verifies the running time of $O(|V_W|)$ for the first **for**-loop of the transformation. The running time of the entire algorithm LDMot becomes $O(|V_W| + |E_W| + T(n))$, by Lemma 3.1 and the assumption that the computation of the cc-partition and function Cov takes $T(n)$ time. The $O(|V_W| + |E_W| + T(n))$ time bound emphasizes once again that the efficiency of LDMot is fully determined by the size of the graph (V_W, E_W) and the time to compute it along with $Cov : V_W \to \mathcal{P}(\mathcal{E})$. Since the time $T(n)$ to compute the graph and the function dominates the time $O(|V_W| + |E_W|)$ to just report both, we may conclude that the $T(n)$-factor dominates the running time of the algorithm LDMot, which may therefore be said to equal $O(T(n))$.

Theorem 3.9 *Let b be a positive constant. Let $\mathcal{B}$ be a constant-complexity robot with reach $\rho_{\mathcal{B}} \leq b \cdot \rho$ moving in a low obstacle density workspace* W *with obstacles $E \in \mathcal{E}$ with minimal enclosing hypersphere radii at least ρ. Furthermore, let $C = \mathrm{W} \times D$ be the configuration space of $\mathcal{B}$. Then, the algorithm LDMot computes the connectivity graph (V_C, E_C) with $|V_C| = O(|V_W|)$ and $|E_C| = O(|E_W|)$ of a cell decomposition of the free space $\mathrm{FP} \subseteq C$ in time $O(T(n))$, where (V_W, E_W) is a cc-partition of* W *and $T(n)$ is the time to compute (V_W, E_W) along with the function $Cov : V_W \to \mathcal{P}(\mathcal{E})$.*

Although the exact performance of the algorithm depends on the ability to find small and efficiently computable cc-partitions, one may, at this stage, expect the method to be rather efficient since the paradigm reduces the problem of finding a decomposition of certain f-cells in an arrangement in f-dimensional configuration space to the problem of finding some constrained partition of the d-dimensional workspace $(d \leq f)$. Besides the dimensional reduction, the hypersurfaces in configuration space have a more complex shape than the obstacles in the workspace that are responsible for the partition constraints. A major part of the next section is devoted to providing small and efficiently computable cc-partitions for workspaces with various kinds of obstacles, showing the usefulness of the approach.

The incorporation of self-collisions has no major implications for the approach outlined above. The constant number of additional constraint hypersurfaces induced by the self-collisions of the constant-complexity robot does not increase the asymptotic complexity of the arrangement inside any cylinder $R \times D$. Therefore, the combinatorial and algorithmic considerations of this section apply without restrictions. The reader is referred to [36] for further details.

4 Small and efficiently computable base partitions

This objective in this section is to find instances of the general paradigm presented in Section 3 for a handful of different settings of the motion planning problem. Besides a universal and nearly-optimal solution for planning in two-dimensional workspaces, we shall consider four different problems in three-dimensional workspaces.

4.1 Arbitrary obstacles in 2-space

We first focus on planar motion planning for an f-DOF free-flying robot $\mathcal{B}$ before we move on to three-dimensional workspaces and obstacles. The configuration space C for this robot $\mathcal{B}$ is the Cartesian product of the 2-dimensional Euclidean workspace W and some $(f-2)$-dimensional space D, hence $C = \mathrm{W} \times D = \mathbb{R}^2 \times D$. For a rigid robot ($f = 3$), the space D equals the one-dimensional rotational interval $[0, 2\pi)$; for free-flying articulated robots ($f \geq 4$), the space D also models the relative placements of the robot's links.

The partition that is proposed below, a vertical decomposition of the arrangement of grown obstacle boundaries, works for any type of obstacles. The first step towards a cc-partition comprises the linear-time computation of all grown obstacle boundaries $\partial G(E, \rho_{\mathcal{B}})$ ($E \in \mathcal{E}$). As a preparation for the next step, each boundary $\partial G(E, \rho_{\mathcal{B}})$ is cut up into (a constant number of) maximal connected, x-monotone arcs having no vertices in their interiors. In addition, each arc from $\partial G(E, \rho_{\mathcal{B}})$ is labeled with E. Lemma 3.8 immediately implies that the resulting $\Theta(n)$ arcs define only $O(n)$ (yet unknown) arc intersections, and additionally subdivide W into $\Theta(n)$ regions with constant-size coverage.

In a second step we compute the vertical decomposition of the arrangement $\mathcal{A}(G)$ of grown obstacle boundaries, by sweeping the plane [5] with the arcs with a vertical line, meanwhile extending walls in upward and downward vertical direction from all $\Theta(n)$ arc endpoints (known in advance) and all $O(n)$ arc intersections (to be determined during the sweep). The extended walls end on the first arc that is hit in the direction of the extension. The walls subdivide the 2-cells of the arrangement into regions bounded by two (possibly degenerate) vertical walls and two arc sections. Hence, the regions in the vertical decomposition of the arrangement $\mathcal{A}(G)$ have constant complexity. Moreover, they inherit the constant-size coverage from the original (enclosing) 2-cell of $\mathcal{A}(G)$. The arcs and their endpoints and pairwise intersections, and the walls and their endpoints, subdivide the plane into $\Theta(n)$ constant-complexity regions: the regions of V_{W}. The set E_{W} of pairs of adjacent vertical decomposition regions is easily seen to have size $\Theta(n)$ as well.

The sweep must not only compute the regions of V_{W}, but also the coverages $Cov(R)$ of the regions $R \in V_{\mathrm{W}}$, and the region adjacencies of E_{W}. Throughout the sweep we maintain as invariant that all regions, coverages, and adjacencies left of the sweep-line are computed. The sweep-line status and event point schedule, both stored in appropriate data structures, facilitate the maintenance of the invariant. The sweep-line status is a top-to-bottom cross-section of the vertical decomposition of $\mathcal{A}(G)$ and the vertical sweep-line. This alternating sequence of arcs and regions accompanied by their coverages is stored in a binary tree. The event point schedule is the ordered sequence (by x-coordinate) of all arc endpoints and the potential intersections of consecutive arcs in the sweep-line status. A priority queue stores the schedule.

The processing of the next event, i.e., the extension of vertical walls from an intersection point or endpoint, marks the end of at most three consecutive regions in the sweep-line status. These regions and their coverages are reported to maintain the invariant. Moreover, the regions are deleted from the sweep-line status and replaced by the (at most three) newly started regions. Appropriate adjustments w.r.t. the separating arcs are made as well. In addition, we must report the adjacencies of the new regions and compute their coverages. (The observation that the coverages of two regions on opposite sides of an arc labeled E differ exactly by $\{E\}$ indicates that the latter computation raises no problems.) Potential intersections of the $O(1)$ newly created pairs of consecutive arcs must be inserted in the event queue. In summary, the processing of an event requires a constant number of searches, deletions, and insertions on a binary tree, and a constant number of insertions in the event queue. All listed operations take $O(\log n)$ time, so the processing of all events, and, hence, the entire sweep, takes $T(n) = O(n \log n)$ time. Substitution of the vertical decomposition in the first step of Algorithm LDMot leads to a cell decomposition of size $\Theta(n)$, computed in $O(n \log n)$ time.

Theorem 4.1 *The low obstacle density motion planning problem in $\mathbb{R}^2$ can be solved in time $O(n \log n)$.*

4.2 Polyhedral obstacles in 3-space

In this subsection, we move on to a three-dimensional workspace where we study a setting of a free-flying robot amidst polyhedral obstacles. The number of algorithms for motion planning problems in a three-dimensional workspace with polyhedral obstacles is limited. The two methods that apply to robots with an arbitrary number $f \geq 3$ of degrees of freedom are the general $O(n^{2^{f+6}})$ cell decomposition algorithm by Schwartz and Sharir [26] and the $O(n^f \log n)$ roadmap method by Canny [8]. More specific results include $O(n^{11})$ [28] and $O(n^6 \log n)$ [15] algorithms for a (5-DOF) ladder among polyhedral obstacles, and an $O(n^{15})$ algorithm [28] for a polyhedral robot in the same environment. This section presents an instance of the algorithm LDMot with running time $O(n^2 \log n)$ for the following class of problems.

> *A constant-complexity robot $\mathcal{B}$ with f degrees of freedom ($f \geq 3$) and reach $\rho_{\mathcal{B}} \leq b \cdot \rho$ moves freely in a low obstacle density workspace $\mathrm{W} = \mathbb{R}^3$ with a collection $\mathcal{E}$ of constant-complexity polyhedral obstacles $E \subseteq \mathrm{W}$ with minimal enclosing sphere radii at least ρ, for some constant $b \geq 0$.*

The problem of finding cc-partitions for three-dimensional Euclidean workspaces is much harder than its two-dimensional equivalent, which is illustrated by the relatively small number of results on partitioning 3D arrangements into constant-complexity subcells like tetrahedra or prisms. Moreover, the existing results (see, for example, papers by Aronov and Sharir [3], Chazelle [9], and De Berg, Guibas, and Halperin [7]) do not apply to arbitrary arrangements but instead only hold for arrangements of planar faces, which makes their application to the arrangement of (arbitrary) grown obstacle boundaries impossible. The two-step approach of first decomposing the workspace into constant-size coverage regions and then refining the regions to constant-complexity regions is likely to lead to $O(n^2)$ regions, as the application of any of the above methods gives $O(n^2)$ subcells when applied to an $O(n)$ complexity arrangement of planar faces. Although a smaller decomposition might be achievable, we are currently unaware of such a decomposition and therefore choose to settle for a cc-partition with $|V_{\mathrm{W}}| = O(n^2)$ and $|E_{\mathrm{W}}| = O(n^2)$. The partition is obtained by computing a constrained triangulation of some polyhedral (outer) approximation of the grown obstacle boundaries, then replacing the triangles by flat non-intersecting tetrahedra, and subsequently applying a full vertical decomposition [7] to the triangular faces of the tetrahedra. The entire computation takes $T(n) = O(n^2 \log n)$ time. The substitution of the flat tetrahedra for the triangles serves as a means of obtaining a provable quadratic number of adjacencies. We discuss each of the steps in more detail below.

Instead of using the grown obstacle boundaries to achieve the decomposition into constant-size coverage regions, we now use the boundaries of the Minkowski sums of the polyhedral obstacles with a cube with side length $2\rho_{\mathcal{B}}$. The Minkowski sums $H(E, \rho_{\mathcal{B}})$ are tight polyhedral outer approximations of the grown obstacles. Let $C_{O,\rho_{\mathcal{B}}}$ be the axis-parallel cube with side length $2\rho_{\mathcal{B}}$ centered at the origin.

Definition 4.2 *Let $\Lambda \in SO(3)$ be some arbitrary rotation matrix, establishing that none of the faces of the cube $\Lambda \cdot C_{O,\rho_{\mathcal{B}}}$ is vertical. Then*

$$H(E, \rho_{\mathcal{B}}) = E \oplus (\Lambda \cdot C_{O,\rho_{\mathcal{B}}}).$$

The computation of $H(E, \rho_{\mathcal{B}})$ from the constant-complexity obstacle E takes constant time. The Minkowski sum $H(E, \rho_{\mathcal{B}})$ encloses E and, by its definition, no point in $H(E, \rho_{\mathcal{B}})$ has a distance more than $\sqrt{3} \cdot \rho_{\mathcal{B}}$ to E. Hence, $H(E, \rho_{\mathcal{B}})$ is a $(\sqrt{3} \cdot \rho_{\mathcal{B}})$-wrapping of E, and by $\rho_{\mathcal{B}} \leq b \cdot \rho$, also a $(b\sqrt{3} \cdot \rho)$-wrapping of E. The additional lower bound of ρ on the minimal enclosing sphere radii of all obstacles in $\mathcal{E}$ makes Theorem 2.7 applicable to the arrangement $\mathcal{A}(H)$ of all Minkowski sum boundaries $\partial H(E, \rho_{\mathcal{B}})$ $(E \in \mathcal{E})$, resulting in Lemma 4.3.

Lemma 4.3 *Let $\mathcal{A}(H)$ be the three-dimensional arrangement of all boundaries $\partial H(E, \rho_{\mathcal{B}})$ $(E \in \mathcal{E})$. Then*

(a) *$\mathcal{A}(H)$ has complexity $O(n)$,*

(b) $|\{E \in \mathcal{E} | H(E, \rho_{\mathcal{B}}) \cap A \neq \emptyset\}| = O(1)$ *for all 3-faces $A \in \mathcal{A}(H)$.*

The set of obstacles E whose Minkowski sums $H(E, \rho_{\mathcal{B}})$ intersect a given region R is a superset of $Cov(R)$, the set of obstacles E whose grown obstacles $G(E, \rho_{\mathcal{B}})$ intersect R, due to the inclusion $G(E, \rho_{\mathcal{B}}) \subseteq H(E, \rho_{\mathcal{B}})$. With the observation we deduce the following interesting corollary from Lemma 4.3(b).

Corollary 4.4 $|Cov(A)| = |\{E \in \mathcal{E} | G(E, \rho_{\mathcal{B}}) \cap A \neq \emptyset\}| = O(1)$ *for all 3-faces $A \in \mathcal{A}(H)$.*

Hence, the $O(n)$ complexity polyhedral arrangement $\mathcal{A}(H)$ subdivides $\mathrm{W} = \mathbb{R}^3$ into regions with constant-size coverage.

A naive and simple computation of $\mathcal{A}(H)$ simply intersects all pairs of constant-complexity faces of Minkowski difference boundaries $\partial H(E, \rho_{\mathcal{B}})$ and stores the (potential) intersection segment with both faces. This computation takes $O(n^2)$, which is sufficient in the light of the complexity of future steps. After that, each face and the segments defined by its intersection with other faces undergo a constrained triangulation. The triangulation is constrained in the sense that it incorporates all intersection edges as triangulation edges and introduces no new vertices. The constrained triangulation can be done by a single sweep of each face f in time $O(m_f \log m_f)$, where m_f is the complexity of the respective face and the corresponding intersection segments. As the cumulative complexity $\sum_f m_f$ equals (asymptotically) the complexity $O(n)$ of the arrangement $\mathcal{A}(H)$, the triangulation of all faces of the arrangement takes $O(n \log n)$ time. The result is a collection $T_{\mathcal{A}(H)}$ of non-intersecting triangles. Although the triangles are non-intersecting they do touch each other, that is, they share edges and vertices. For future purposes, we label each triangle $t \in T_{\mathcal{A}(H)}$ with the appropriate obstacle E to indicate that t belongs to the boundary $\partial H(E, \rho_{\mathcal{B}})$. The decomposition of the workspace by the triangulated arrangement $\mathcal{A}(H)$ is not a cc-partition, as the resulting regions may have more than constant complexity.

An elegant way to overcome future problems in the analysis of the number of adjacencies is to replace all triangles of $T_{\mathcal{A}(H)}$ by tetrahedra that are sufficiently flat to prevent them from intersecting. The tetrahedra have the initial triangles of $T_{\mathcal{A}(H)}$ as one of their faces. The triangular faces of the tetrahedra constitute a set $F(T_{\mathcal{A}(H)}) \supseteq T_{\mathcal{A}(H)}$ with $|F(T_{\mathcal{A}(H)})| = 4 \cdot |T_{\mathcal{A}(H)}| = O(n)$. The set $F(T_{\mathcal{A}(H)})$ consists of non-intersecting though touching triangles, which now have the simple but beneficial property that one of their sides faces the interior of a tetrahedron τ (consisting of four triangles from $F(T_{\mathcal{A}(H)})$). The property will come to our help when we analyze the complexity of the full vertical decomposition of the triangles.

We create the tetrahedra of $F(T_{\mathcal{A}(H)})$ as follows. Recall that $T_{\mathcal{A}(H)}$ is a set of triangles that may share a vertex or an edge but do not intersect each others' interiors. Let ϵ be the minimum distance between any pair of disjoint (non-touching) triangles. Furthermore, let γ_1 be the minimum over all dihedral angles between pairs of (touching) triangles that share an edge and γ_2 be the minimum over all angles between the supporting plane of a triangle $t \in T_{\mathcal{A}(H)}$ and an edge of another triangle t' incident to a vertex of t. We define $\gamma = \min(\gamma_1, \gamma_2)$. We construct a set $F(T_{\mathcal{A}(H)})$ by applying the following procedure to every $t \in T_{\mathcal{A}(H)}$. Let v_1, v_2, v_3 be the vertices of t.

1. The planes π_1, π_2, π_3 through v_1, v_2, v_3 that make a positive angle $\gamma/2$ with the top side (facing $z = \infty$) of t intersect in a point v. If the distance from v to t is strictly less than ϵ, then the tetrahedron is defined by the vertices v, v_1, v_2, v_3.

2. If the distance from v to t is at least ϵ, then we take the half-line h through v and perpendicular to and ending on t. The tetrahedron is defined by v_1, v_2, v_3 and the unique point v' on the half-line h with distance $\epsilon/2$ to t.

It is not hard to see that the application of the above process results in a collection of tetrahedra with disjoint interiors. The faces of these tetrahedra constitute $F(T_{\mathcal{A}(H)})$. The triangles of $F(T_{\mathcal{A}(H)})$ partition the workspace W with the obstacles $\mathcal{E}$ into regions with constant-size coverage. This follows directly from the construction of $F(T_{\mathcal{A}(H)})$ from the triangles of $T_{\mathcal{A}(H)}$, which already define a partition of into regions with constant-size coverage. The addition of disjoint triangles to the partition can only lead to a refinement of the regions into smaller regions, with smaller or equally-sized coverages.

De Berg, Guibas, and Halperin [7] present an algorithm for computing a full vertical decomposition of an arrangement of triangles in general position in 3-space. The general position of the obstacles in $\mathcal{E}$ and the rotated cube $\Lambda \cdot C_{O,\rho_{\mathcal{B}}}$ establish that the triangles of $T_{\mathcal{A}(H)}$ and $F(T_{\mathcal{A}(H)})$ are in general position in the sense that no triangle is vertical, no edge is parallel to a coordinate axis, and no two edges lie in a vertical plane unless they coincide. The computation of the full vertical decomposition of the arrangement of non-intersecting triangles from $F(T_{\mathcal{A}(H)})$ proceeds in two steps. In the first step, walls are extended in vertical ($z-$)direction from all triangle edges. The walls end upon hitting another triangle. The fact that we deal with sets of touching triangles requires some simple additional bookkeeping to prevent multiple extensions of equivalent walls. The result is an intermediate decomposition of the arrangement into regions bounded by vertical walls and by a single triangle from above and from below. The second step refines each region in the intermediate decomposition by extending additional walls parallel to the (x, z)-plane, to obtain a decomposition into regions bounded by six (possibly degenerate) planar faces. The number of (x, z)-parallel walls necessary to achieve this refinement of a region is of the same order of magnitude as the complexity of that region. Like the wall extended in the first phase, the walls added during the refinement phase do not intersect triangles. The resulting full vertical decomposition has complexity $O(n^2)$. Its computation takes $O(n^2 \log n)$ time. The constant-complexity regions constitute the set V_{W}. The regions inherit the constant-size coverage from the 3-cells of the arrangement of triangles. In conclusion, the set V_{W} is a cc-partition of the workspace of size $O(n^2)$ and is computable in time $O(n^2 \log n)$ time.

The representation of the full vertical decomposition consists of: (i) all walls extended from triangle plus all additional walls parallel to the (x, z)-plane, and (ii) for each side of a triangle $t \in F(T_{\mathcal{A}(H)})$, the arrangement of ending walls. The algorithm stores the walls and the triangle arrangements in a quad-edge structure [11] to facilitate future navigating through the decomposition.

Let us now bound the size of the set $E_{\mathrm{W}} = \{(R, R') \in V_{\mathrm{W}} \times V_{\mathrm{W}} | \partial R \cap \partial R' \neq \emptyset\}$. The complexities of the triangle arrangements are crucial to the analysis of the adjacencies, so we first study these complexities in more detail. As the complexity of the entire full vertical decomposition is $O(n^2)$, the cumulative complexity of all triangle arrangements is $O(n^2)$ as well. Each triangle $t \in F(T_{\mathcal{A}(H)})$ has a side facing the interior of the tetrahedron τ it belongs to and a side facing outward. Walls that are extended outside τ are unable to penetrate τ, as each wall ends upon hitting (the outside of) one of the triangular faces of τ. The walls inside τ must therefore either be extended from one of the six edges of τ, or added during the subsequent refinement. Clearly, the number of walls inside τ is constant. As a result, the complexity m_t of the arrangement of the $O(1)$ ending walls on the inward-facing side of t is constant, for all t. The complexity n_t of the arrangement on the outward-facing side of a single triangle t, however, can be as high as $O(n^2)$. The numbers of 2-faces and adjacencies of pairs of 2-faces in a triangle arrangement are of the same order of magnitude as the complexity of the arrangement: $O(m_t)$ for the inward-facing side and $O(n_t)$ for the outward-facing side.

Two adjacent regions share a common boundary that is either embedded in a triangle or embedded in a vertical wall. Each non-empty intersection of 2-faces in arrangements on either side of a triangle represents an adjacency of the first type. The number of non-empty intersections on opposite sides of a triangle t is $O(m_t \cdot n_t) = O(n_t)$ due to $m_t = O(1)$. Hence, the number of adjacencies in E_{W} of the first type equals $\sum_t O(n_t) = O(n^2)$. Notice that it is crucial to replace the triangles by flat tetrahedra. If we would apply the full vertical decomposition to the triangles of $T_{\mathcal{A}(H)}$ themselves, the arrangements on the two sides of the triangle can easily *both* have complexity $O(n^2)$, leading to an unclear number of adjacencies of the type just considered. To find the number of adjacencies of the second type, note that two adjacent regions whose common boundary is part of a vertical wall have adjacent floors or adjacent ceilings in a triangle arrangement. Hence, the total number of adjacencies of 2-faces in all triangle arrangements supplies an upper bound on the number of pairs of regions in V_{W} that share a vertical face. The total number of adjacencies of 2-faces on a triangle t is $O(m_t + n_t) = O(n_t)$. Hence, the number of adjacencies of the second type equals $\sum_t O(n_t) = O(n^2)$ as well. Lemma 4.5 summarizes the bounds on the sizes of V_{W} and E_{W}.

Lemma 4.5 *V_{W} is a cc-partition of size $O(n^2)$ of $\mathrm{W} = \mathbb{R}^3$ with the polyhedral obstacles $\mathcal{E}$; $E_{\mathrm{W}} = \{(R, R') \in V_{\mathrm{W}} \times V_{\mathrm{W}} | \partial R \cap \partial R' \neq \emptyset\}$ has size $O(n^2)$.*

After applying the $O(n^2 \log n)$ vertical decomposition algorithm, we traverse the $O(n^2)$ constant-complexity regions of the decomposition using the quad-edge structure, starting from an arbitrary region. The aim of the traversal is to extract explicit descriptions of the regions in V_{W}, report the region adjacencies of E_{W}, and compute the coverages $Cov(R)$ of the regions $R \in V_{\mathrm{W}}$. The latter part of the computation deserves some additional explanation. Instead of attempting to compute the constant-size coverages $Cov(R) = \{E \in \mathcal{E} | G(E, \rho_{\mathcal{B}}) \cap R \neq \emptyset\}$ directly, we first compute the constant-cardinality supersets $\{E \in \mathcal{E} | H(E, \rho_{\mathcal{B}}) \cap R \neq \emptyset\}$ and then eliminate from each set - in constant time - all members E with $G(E, \rho_{\mathcal{B}}) \cap R = \emptyset$. The constant cardinality of these supersets follows from the property that each region $R \in V_{\mathrm{W}}$ is a subset of a 3-face A of the arrangement $\mathcal{A}(H)$, which satisfy $|\{E \in \mathcal{E} | H(E, \rho_{\mathcal{B}}) \cap A \neq \emptyset\}| = O(1)$, by Lemma 4.3. The necessary data for the computation of the sets $\{E \in \mathcal{E} | H(E, \rho_{\mathcal{B}}) \cap R \neq \emptyset\}$ are available from the decomposition, contrary to the data for the computation of $Cov(R)$. Throughout the traversal of the decomposition we use the fact that adjacent regions are intersected (or actually enclosed) by the same set of Minkowski sums $H(E, \rho_{\mathcal{B}})$ unless their common boundary is contained in a triangle t that is part of some Minkowski sum boundary $\partial H(E, \rho_{\mathcal{B}})$, in which case the sets of intersecting Minkowski sums differ by exactly $\{E\}$: the label of t. The traversal of the $O(n^2)$ regions in the decomposition requires time proportional to the number of regions. As a consequence, the time $T(n)$ to compute the cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the function Cov is dominated by the running time of the vertical decomposition algorithm: $O(n^2 \log n)$.

Substitution of the computation of $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and Cov outlined above for the first step of the algorithm LDMot yields a motion planning algorithm with running time $O(n^2 \log n)$. The algorithm decomposes the free space of a robot amidst polyhedral obstacles into $O(n^2)$ subcells of constant-complexity, defining $O(n^2)$ pairwise adjacencies.

Theorem 4.6 *The low obstacle density motion planning problem amidst polyhedral obstacles in $\mathbb{R}^3$ can be solved in time $O(n^2 \log n)$.*

The gap between the (linear) complexity of the free space and the (quadratic) size of the connectivity graph of the FP decomposition shows that the cell decomposition is not optimal. An interesting open problem is therefore to attempt to bridge the gap by exploring alternative cc-partitions of the workspace.

4.3 Arbitrary obstacles in 3-space

We now relax the restrictions on the obstacles and allow them to be arbitrarily shaped. The only general methods that could solve such a problem are those by Schwartz and Sharir [25] and Canny [8]. Here, it is shown that an instance of the algorithm LDMot for a bounded-size robot in a low density workspace with arbitrary obstacles exists with running time $\Theta(n^3)$, independent of the actual number f of degrees of freedom of the robot. The setting is fixed by the following description.

> *A constant-complexity robot $\mathcal{B}$ with f degrees of freedom ($f \geq 3$) and reach $\rho_{\mathcal{B}} \leq b \cdot \rho$ moves freely in a low obstacle density workspace $\mathrm{W} = \mathbb{R}^3$ amidst a collection $\mathcal{E}$ of constant-complexity obstacles $E \subseteq \mathrm{W}$ with minimal enclosing sphere radii at least ρ, for some constant $b \geq 0$.*

The main implication of the arbitrary shape of the obstacles is that they can no longer be tightly wrapped by some polyhedron of constant complexity. A consequence is that we are no longer able to construct a low complexity arrangement in a first decomposition step which partitions the workspace into regions with constant-size coverage. Instead, we propose a simple direct cc-partition of the workspace leading to a cubic number of regions.

The axis-parallel bounding boxes $B(G(E, \rho_{\mathcal{B}}))$ of the grown obstacles $G(E, \rho_{\mathcal{B}})$ ($E \in \mathcal{E}$) partition the workspace with the obstacles $\mathcal{E}$ into regions with constant coverage. The arrangement of boxes has complexity $O(n^3)$. Unfortunately, the 3-cells of the arrangement may have more than constant complexity. If, however, we replace the arrangement of bounding boxes by the arrangement of the supporting planes of all bounding box faces, then we obtain rectangloid 3-cells without increasing the asymptotic worst-case complexity of the arrangement: the cc-partition of the workspace by the supporting planes has complexity $\Theta(n^3)$.

Each obstacle $E \in \mathcal{E}$ contributes six planes to the arrangement defining the cc-partition. The constant complexity of the obstacle E allows us to compute the supporting planes $x = x'$, $x = x''$, $y = y'$, $y = y''$, $z = z'$, and $z = z''$ of the grown obstacle $G(E, \rho)$ in constant time. For simplicity, we assume that none of the supporting planes is tangent to a second grown obstacle $G(E', \rho_{\mathcal{B}})$ ($E' \neq E$). After having computed all $2n$ planes parallel to the (y, z)-plane, we sort the resulting planes by increasing x-coordinates, to obtain a sequence $x_1 < \ldots < x_{2n}$. The sequence partitions the real line into $2n + 1$ intervals X_h ($0 \leq h \leq 2n$) with $X_0 = (-\infty, x_1]$, $X_h = [x_h, x_{h+1}]$ for all $1 \leq h < 2n$, and $X_{2n} = [x_{2n}, +\infty)$. A similar treatment of the planes parallel to the (x, z)-plane and (x, y)-plane results in sequences $y_1 < \ldots < y_{2n}$ and $z_1 < \ldots < z_{2n}$ and two partitions of the real line into intervals Y_i ($0 \leq i \leq 2n$) and Z_j ($0 \leq j \leq 2n$) respectively. The three ordered sequences define a cc-partition graph consisting of a set V_{W} of regions

$$V_{\mathrm{W}} = \{ X_h \times Y_i \times Z_j \mid 0 \leq h, i, j \leq 2n \}.$$

Each region has six neighbors with each of which it shares one of its six rectangular faces.

Lemma 4.7 *V_{W} is a cc-partition of size $\Theta(n^3)$ of W with the obstacles $\mathcal{E}$; $E_{\mathrm{W}} = \{(R, R') \in V_{\mathrm{W}} \times V_{\mathrm{W}} | \partial R \cap \partial R' \neq \emptyset\}$ has size $\Theta(n^3)$.*

Proof: V_{W} and E_{W} are easily seen to have size $\Theta(n^3)$. The remaining task is to prove that the constant-complexity regions of V_{W} have constant-size coverage.

The structure of the partition by the supporting planes of the bounding boxes of the grown obstacles is such that an arbitrary rectangloid region $R \in V_{\mathrm{W}}$ either lies in the exterior of all bounding boxes, in which case it has empty coverage, or it lies entirely in the interior of a number of bounding boxes of grown obstacles. Let, in the latter case, D be the set of all obstacles E for which $R \subseteq B(G(E, \rho_{\mathcal{B}}))$. Let E^- be the obstacle in D with the smallest minimal enclosing sphere radius, say, ρ^-. We first prove that no obstacles with minimal sphere radii smaller than ρ^- belong to $Cov(R)$. Assume, for a contradiction, that E^* has minimal enclosing sphere radius $\rho^* < \rho^-$ and satisfies $E^* \in Cov(R)$. By the definition of coverage, this means that, $R \cap G(E^*, \rho_{\mathcal{B}}) \neq \emptyset$. But then, since $G(E^*, \rho_{\mathcal{B}}) \subseteq B(G(E^*, \rho_{\mathcal{B}}))$, also $R \cap B(G(E^*, \rho_{\mathcal{B}})) \neq \emptyset$. So, E^* must belong to D, violating the assumption that E^- is the obstacle in D with the smallest minimal enclosing sphere radius.

From $E^- \in D$ it follows that $R \subseteq B(G(E^-, \rho_{\mathcal{B}}))$. The minimal enclosing hypersphere radius ρ^- of E^- implies that the length of none of the sides of $B(G(E^-, \rho_{\mathcal{B}}))$ exceeds $2\rho^- + 2\rho_{\mathcal{B}} \leq 2\rho^- + 2b\rho \leq (2b + 2)\rho^-$. As a result, the length of none of the sides of $R \subseteq B(G(E^-, \rho_{\mathcal{B}}))$ exceeds $(2b + 2)\rho^-$ as well. An obstacle E' with minimal enclosing sphere radius $\rho' \geq \rho^-$ whose corresponding grown obstacle $G(E', \rho_{\mathcal{B}})$ intersects R, must itself intersect the enclosing rectangloid $R^G \supseteq R$, obtained by growing R by $\rho_{\mathcal{B}} \leq b\rho \leq b\rho^-$ in all (six) axis-parallel directions, so $R^G = R \ominus C_{O,\rho_{\mathcal{B}}}$. The edges of R^G have length at most $(4b + 2)\rho^-$. By Property 2.2, the number of obstacles E' with minimal enclosing sphere radius $\rho' \geq \rho^-$ intersecting the rectangloid region R^G is bounded by a constant, and hence the number of grown obstacles $G(E', \rho_{\mathcal{B}})$ intersecting R is bounded by a constant, meaning that $|Cov(R)| = O(1)$. $\square$

The single algorithmic issue to be solved concerns the computation of the coverage $Cov(R)$ of each region $R \in V_{\mathrm{W}}$, because the regions of V_{W} and the adjacencies of E_{W} can be trivially extracted in time $\Theta(n^3)$ from the the three ordered sequences of planes. Instead of taking a single region $R \in V_{\mathrm{W}}$ and computing all grown obstacles $G(E, \rho_{\mathcal{B}})$ that intersect it, we choose a more or less inverse approach here: we take a grown obstacle region $G(E, \rho_{\mathcal{B}})$ and compute all regions $R \in V_{\mathrm{W}}$ that are intersected by it, and add E to all corresponding sets $Cov(R)$ under construction. In other words, we want to determine all regions R with $Cov(R) \ni E$. The approach is to identify a single region R intersected by $G(E, \rho_{\mathcal{B}})$ and then use this region as a basis for searching the adjacency graph E_{W} to find the entire connected set of regions intersected by $G(E, \rho_{\mathcal{B}})$. The correctness of the approach relies on the connectedness of $G(E, \rho_{\mathcal{B}})$, which is implied by the connectedness of E.

To find an arbitrary region R intersected by $G(E, \rho_{\mathcal{B}})$, we take a point $p \in G(E, \rho_{\mathcal{B}})$. Next, we perform a point location query with $p = (p_x, p_y, p_z)$ in the partition V_{W}, which, by the orthogonality of the partition can be decomposed into three binary searches to find $X_h \ni p_x$, $Y_i \ni p_y$, and $Z_j \ni p_z$ in time $O(\log n)$. The Cartesian product $X_h \times Y_i \times Z_j$ contains the point p, and, hence, intersects $G(E, \rho_{\mathcal{B}})$.

In preparation for a second phase, we create a set Ev yet consisting of just one element: R. We repeatedly extract an element R from Ev for further processing, until Ev is empty. We compute the six neighbors of R and verify for each neighbor R' if R' intersects $G(E, \rho_{\mathcal{B}})$. If this is the case and the neighbor has not been treated yet (which can be tested by marking the regions visited), then we add E to $Cov(R')$ and the neighbor R' itself to Ev.

The above search through the regions considers a superset of the collection of m_E regions $R \in V_{\mathrm{W}}$ satisfying $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$. More precisely, it also considers all regions adjacent to regions R with $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$. Still, the total number of regions that are considered is $O(m_E)$. As the amount of work per region is constant, the total time spent in the search is

$O(m_E)$.

It remains to bound the number m_E of regions $R \in V_{\mathrm{W}}$ with $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$. For each region R with $R \cap G(E, \rho_{\mathcal{B}}) \neq \emptyset$, we add E to $Cov(R)$. Hence, the sum of all m_E over all grown obstacles $G(E, \rho_{\mathcal{B}})$ equals the sum of all $|Cov(R)|$ over all regions R. As $|Cov(R)| = O(1)$ for all $R \in V_{\mathrm{W}}$, the latter sum amounts to $O(n^3)$. As a result, the sum of all m_E equals $O(n^3)$, and, hence, the cumulative search time is $O(n^3)$. The n point location queries (in the orthogonal cc-partition) to identify the starting regions require additional $O(n \log n)$ in total. As a result, the cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the corresponding function Cov can be computed in $T(n) = \Theta(n^3)$ time.

The substitution of the computation of the cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the function Cov in the first step of the algorithm LDMot leads, by Theorem 3.9, to a motion planning algorithm that decomposes FP into constant-complexity subcells in time $T(n) = \Theta(n^3)$. The number of subcells and subcell adjacencies is in the worst case of the same order of magnitude as the number of regions and adjacencies in the cc-partition.

Theorem 4.8 *The low density motion planning problem amidst arbitrary obstacles in* $\mathbb{R}^3$ *can be solved in time* $\Theta(n^3)$.

4.4 Similarly-sized obstacles in 3-space

The motion planning algorithm in the previous subsection has a relatively high running time compared to the free space complexity. It is therefore interesting to see what realistic additional assumptions may lead to a relevant improvement of the performance. This subsection shows an interesting example of such a realistic assumption, namely that the obstacles have comparable sizes. The assumption is realistic because in many practical situations, the largest obstacle in the workspace will not be more than a constant factor bigger than the smallest obstacle. The addition of the assumption to the mildly constrained setting of the previous section leads to a surprising improvement of the performance. The resulting motion planning algorithm computes an optimal $O(n)$ cell decomposition in nearly-optimal $O(n \log n)$ time. The following problem statement fixes the setting of the results in this section.

> *A constant-complexity robot* $\mathcal{B}$ *with* f *degrees of freedom* ($f \geq 3$) *and reach* $\rho_{\mathcal{B}} \leq b \cdot \rho$ *moves freely in a low obstacle density workspace* $\mathrm{W} = \mathbb{R}^3$ *with a collection* $\mathcal{E}$ *of constant-complexity obstacles* $E \subseteq \mathrm{W}$ *with minimal enclosing sphere radii in the range* $[\rho, u\rho]$, *for some constants* $b \geq 0$ *and* $u \geq 1$.

The bounded ratio between the size of the smallest and largest obstacle in $\mathcal{E}$ provides the opportunity of a simple and structured cc-partition consisting of axis-parallel rectangloid regions, each being the union of one or more cubes of the form $[h\rho, h\rho + \rho] \times [i\rho, i\rho + \rho] \times [j\rho, j\rho + \rho]$, where $h, i, j \in \mathbf{Z}$. The regions are of one of three types: type 1 regions are single cubes, type 2 regions have width and height ρ, and type 3 regions are unbounded in both z-directions. Type 2 and type 3 regions have empty coverage. The number of regions of each type in the subdivision is bounded by $O(n)$. Moreover, the number of adjacencies is equally low: $O(n)$. The rectangloid subdivision is as such an optimal cc-partition of the workspace.

The basic idea behind the cc-partition is to embed all grown obstacles $G(E, \rho_{\mathcal{B}})$ in cubes $[h\rho, h\rho + \rho] \times [i\rho, i\rho + \rho] \times [j\rho, j\rho + \rho]$, where $h, i, j \in \mathbf{Z}$. The cubes that intersect at least at least one grown obstacle $G(E, \rho_{\mathcal{B}})$ are the type 1 regions, which together constitute the set V_1. On the one hand, Property 2.2 implies that these cubic regions are small enough to certify constant-size coverage, while, on the other hand, the bound on the size of the obstacles guarantees that each grown obstacle can be embedded in a constant number of cubic regions with pair-wise disjoint interiors. As a result, the set V_1 consists of $O(n)$ regions with constant-size coverage. The computation of V_1 proceeds in a straightforward manner. First, the set of cubes intersecting the grown obstacle $G(E, \rho_{\mathcal{B}})$ is computed for each obstacle $E \in \mathcal{E}$. Then, we simply sort the $O(n)$ cubes lexicographically to eliminate multiple copies of a single cube. Hence, the computation of V_1 can be accomplished in time $O(n \log n)$.

Let $Pr(V_1)$ be the set of projections of cubes $R \in V_1$ on their first two coordinates. All adjacent empty-coverage cubes in one column $C \times \mathbb{R}$, where $C \in Pr(V_1)$, are merged into a single (possibly semi-infinite) constant-complexity type 2 region of width and height ρ. By construction, the number of type 2 regions in a single column cannot exceed the number of type 1 regions in that column by more than one. As a result, the set V_2 of type 2 regions has $O(n)$ members. Using the ordered sequence of type 1 regions, we can compute the type 2 regions in time proportional to the length of the sequence, hence $O(n)$. The regions of $V_1 \cup V_2$ jointly partition the union of all columns $C \times \mathbb{R}$, with $C \in Pr(V_1)$.

As a final step it suffices to subdivide the planar complement of the $O(n)$ squares $C \in Pr(V_1)$ in the (x, y)-plane into regions of constant complexity. The orthogonal liftings into the z-dimension of the planar subdivision regions jointly partition the complement of the columns $C \times \mathbb{R}$ into constant-complexity regions. The liftings, which are unbounded in the z-direction, are appointed to be the type 3 regions. A straightforward subdivision of the planar complement of the squares is a vertical decomposition. The vertical decomposition results in $O(n)$ regions and takes time $O(n \log n)$. As a consequence, the set V_3 of type 3 regions has $O(n)$ elements.

The results obtained so far show that the regions of $V_1 \cup V_2 \cup V_3$ have constant-complexity and constant-size coverage. In addition, the regions collectively partition $\mathbb{R}^3$, which makes $\mathrm{W} = V_1 \cup V_2 \cup V_3$ an adequate choice for a cc-partition of the workspace $\mathrm{W} = \mathbb{R}^3$.

Lemma 4.9 *$V_{\mathrm{W}} = V_1 \cup V_2 \cup V_3$ is a cc-partition of size $O(n)$ of* W *with the obstacles $\mathcal{E}$; the computation of V_{W} takes $O(n \log n)$ time.*

Lemma 4.10 states an upper bound on the number of region adjacencies in the cc-partition. The reader is referred to [36] for a proof of the result.

Lemma 4.10 *$E_{\mathrm{W}} = \{(R, R') \in V_{\mathrm{W}} \times V_{\mathrm{W}} | \partial R \cap \partial R' \neq \emptyset\}$ has size $O(n)$.*

The cc-partition of $\mathrm{W} = \mathbb{R}^3$ by the regions of V_{W} has a recursive structure which turns out to be useful in the computation of the region adjacencies. At the upper level, the workspace W is divided into slices, separated by planes $x = h\rho$ ($h \in \mathbf{Z}$). A slice is either a region from V_3 or it is divided into levels by planes $y = i\rho$ ($i \in \mathbf{Z}$) and has width ρ. A level is either a region from V_3 or it is a column of width and height ρ, consisting of type 1 and 2 regions. The recursive structure of the subdivision allows for a lexicographical ordering - computable in time $O(n \log n)$ - of its regions: slice-by-slice by increasing x, level-by-level by increasing y within a single slice, and by increasing z within a single level. The ordering makes it easy to distinguish three different types of adjacencies. Two adjacent regions are x-adjacent if they lie in different (subsequent) slices; two adjacent regions are y-adjacent if they lie in different (subsequent) levels of a single slice; two adjacent regions in a single level (a column) are z-adjacent. The z-adjacencies are straightforwardly computable in time $O(n)$, as any z-adjacent pair appears consecutively in the ordered sequence of regions. To compute the y-adjacencies we consider all - at most $O(n)$ - pairs of subsequent levels within a single slice. For each such pair, we simultaneously scan the two levels from $z = -\infty$ to $z = \infty$ to report all adjacencies induced by their regions. The scan takes time proportional to the number of adjacencies. To compute the y-adjacencies we consider all - at most $O(n)$ - pairs of adjacent levels in subsequent slices. For each such pair, a simultaneous scan of the two levels yields all adjacencies. In summary, the computation of the adjacencies takes, given the ordered sequence of regions, time proportional to the number of adjacencies, hence, $O(n)$.

To compute the coverage of the regions of V_{W}, we borrow the ideas from Section 4.3. Thus, the approach is to compute for each obstacle $E \in \mathcal{E}$ all regions $R \in V_{\mathrm{W}}$ (in fact $R \in V_1$) intersected by $G(E, \rho_{\mathcal{B}})$. One arbitrary region $R \in V_{\mathrm{W}}$ intersecting the grown obstacle can be determined in $O(\log n)$ time, using the ordered sequence of regions. Starting from that region, the other $O(1)$ regions intersected by the grown obstacle are identified in constant time, using E_{W}. The approach amounts to $O(n \log n)$ for computing Cov.

If we substitute the above computation of the cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the corresponding function Cov for the first step of algorithm LDMot, then we obtain an efficient algorithm for computing a cell decomposition of the free space. The algorithm yields a decomposition into $O(n)$ constant-complexity cells in time $T(n) = O(n \log n)$ by Theorem 3.9.

Theorem 4.11 *The low density motion planning problem amidst similarly-sized arbitrary obstacles in* $\mathbb{R}^3$ *can be solved in time* $O(n \log n)$.

4.5 Planar motion in 3-space

With the present technological state-of-the-art, one rarely encounters free-flying three-dimensional robots in industrial environments. Instead, many robots move in three-dimensional workspaces amidst spatial obstacles while their motion is confined to a (planar) workfloor. An example of such a setting is a vacuum cleaner moving in a room in which objects hang from the ceiling and stand on the floor [37]. Sometimes, the nature and positions of the obstacles do not allow to reduce such problem to purely planar motion planning. The vacuum cleaner, for example, can easily pass under a table. An approach to solve the problem by projecting the vacuum cleaner and the obstacles onto the floor and then finding a (planar) path for the projected vacuum cleaner amidst the projected obstacles would forbid such paths. In this subsection we study a general formulation of the type of problem outlined above in the context of low obstacle density workspaces.

The motion of the robot $\mathcal{B}$ in the three dimensional low obstacle density workspace is constrained by the assumption that a specific point p in $\mathcal{B}$ is restricted to a workfloor. Without loss of generality, we assume that this workfloor is $\mathrm{W}_0 = \{(x, y, 0) \in \mathrm{W}\}$. For convenience, we choose the robot's reference point $O \in \mathcal{B}$ to be equal to the point p that is restricted to the workfloor. Hence, $O[Z] \in \mathrm{W}_0$ for all placements $Z \in C$ of the robot $\mathcal{B}$. The following problem statement fixes the setting.

> *A constant-complexity robot* $\mathcal{B}$ *with* f *degrees of freedom* ($f \geq 2$) *and reach* $\rho_{\mathcal{B}} \leq b \cdot \rho$ *moves with some point* $O \in \mathcal{B}$ *restricted to the plane* $\mathrm{W}_0 = \{(x, y, 0) \in \mathrm{W}\}$ *in a low obstacle density workspace* $\mathrm{W} = \mathbb{R}^3$ *amidst a collection* $\mathcal{E}$ *of constant-complexity obstacles* $E \subseteq \mathrm{W}$ *with minimal enclosing sphere radii at least* ρ, *for some constant* $b \geq 0$.

Note that the problem statement does not restrict the robot to rotate around an axis perpendicular to the workfloor only (as for the vacuum cleaner). The robot is allowed to rotate arbitrarily and can have more degrees of freedom. (An example of such a robot is a moon vehicle, equipped with several arms to grasp stones, etc.)

Planar motion planning in 3-space is sometimes referred to as two-and-a-half-dimensional motion planning, for understandable reasons. A solution to this type of path-finding for a polyhedral robot amidst polyhedral obstacles is given by Wentink and Schwarzkopf in [37]. Their algorithm, which is a generalization of the boundary cell decomposition algorithm by Avnaim, Boissonnat, and Faverjon [4], runs in time $O(n^3 \log n)$. Under our assumptions, the free space can be decomposed into $O(n)$ simple subcells in time $O(n \log n)$, using the general ideas of Section 3.1, yielding an $O(n \log n)$ motion planning algorithm.

We choose the configuration space to be

$$C = \mathrm{F} \times D = \mathbb{R}^2 \times D,$$

where F is the projection of W_0 onto the first two (non-zero) coordinates ($\mathrm{F} = \mathbb{R}^2$) and D is again some rest-space. Any point $Z = (Z_{\mathrm{F}}, Z_D) = ((x,y), Z_D)$, with $Z_{\mathrm{F}} \in \mathrm{F}, Z_D \in D$, in configuration space corresponds to a placement of $\mathcal{B}$ in which its reference point is positioned at $(x,y,0)$ in the workspace W. A possible rest-space for a vacuum cleaner would be $D = [0, 2\pi)$. The configuration space C formulated above makes the application of the tailored paradigm from Section 3.2 for solving the motion planning problem impossible, as the configuration space is not a superspace of the robot's workspace. As a consequence, we have to rely on the more general ideas of Section 3.1 and try to find a valid base partition of the subspace F of C. Fortunately, this turns out to be a rather easy task, basically because the beneficial low density properties of the workspace W are inherited by the subspace $\mathrm{W}_0 = \mathrm{F} \times \{0\}$. The claim is that the vertical decomposition - equivalent to the one in Subsection 4.1 - of (the projection of) the planar arrangement obtained by intersecting the three-dimensional grown obstacle boundaries and the workfloor W_0 is a valid base partition of F. Below we give an informal explanation for the correctness of this claim. The reader is referred to [36] for a more thorough explanation.

The grown obstacle boundaries partition the workspace W into regions of constant-size coverage. When the robot's reference point is restricted to such a region, the robot can only touch a constant number of different obstacles. Clearly, this property is inherited by the two-dimensional subsets of these regions that are defined by the intersection of the workfloor and the grown obstacle boundaries. The three-dimensional linear complexity arrangement $\mathcal{A}(G)$ defines a linear complexity arrangement in the workfloor. The 2-faces of the arrangement have constant-size three-dimensional coverage. Now, let $R \subseteq \mathrm{F}$ be the projection of such a face, The obstacle feature Φ defining a constraint hypersurface $f_{\phi,\Phi}$ intersecting the configuration space cylinder $R \times D$ must clearly belong to $Cov(R \times \{0\})$, otherwise $\mathcal{B}$'s feature could not be touching it while its reference point is restricted to $(x,y,0) \in R \times \{0\}$. The constant size of this set implies that there can only be a constant number of such constraint hypersurfaces. Hence, the linear complexity arrangement partitions the base space F into regions R such that the liftings $R \times D$ are intersected by only $O(1)$ constraint hypersurfaces. A planar vertical decomposition turns the partition into a valid base partition.

The computation of the base partition is straightforward. The intersections of the grown obstacles and the workfloor are obtained in linear time. A plane sweep yields the regions, adjacencies, and three-dimensional coverages in $O(n \log n)$ time (see Subsection 4.1). The base partition is of size $O(n)$. The transformation algorithm from Subsection 3.1 transforms the base partition in $O(n)$ time into a cell decomposition of the configuration space of size

$O(n)$. The three-dimensional coverages of the two-dimensional regions guarantee that step 1 of the first loop can be executed efficiently, as they provide a constant-cardinality superset of the set of constraint hypersurfaces intersecting the configuration space cylinders.

Theorem 4.12 *The low density motion planning problem for a robot moving on a planar workfloor amidst arbitrary obstacles in* $\mathbb{R}^3$ *can be solved in time* $O(n \log n)$.

5 Conclusion

We have studied the motion planning problem for a constant-complexity robot $\mathcal{B}$ with f degrees of freedom in a low obstacle density workspace with n constant-complexity obstacles $E \subseteq \mathbb{R}^d$, for some constants $d, f \geq 0$. The reach $\rho_{\mathcal{B}}$ of the robot $\mathcal{B}$ is assumed to be bounded from above by a constant multiple $b \geq 0$ of ρ, where ρ is a lower bound on the diameter of any obstacle E. The mild assumptions provide a realistic framework for many practical motion planning problems. The complexity of the free space for problems that satisfy the assumptions was proven to be $O(n)$ [33], whereas the complexity can easily be as high as $\Omega(n^f)$ when both assumptions are dropped.

Besides having a low combinatorial complexity, the free space for a motion planning problem that fits in our framework also has a beneficial structure. The structure allows for a decomposition of the configuration space $C = \mathrm{W} \times D$ into cylinders, with bases in the workspace W such that the free space part of every cylinder has constant-complexity. This reduces the problem of finding a cell decomposition of the free space to the problem of finding some constrained decomposition of the lower-dimensional workspace. A uniform sequence of operations then suffices to transform the workspace partition into a cell decomposition of the free space of asymptotically equal size. The running time of the entire paradigm is determined by the time to compute the workspace partition.

Optimal $O(n)$ size workspace partitions exist for three out of five practical instances of the low obstacle density motion planning problem studied in this paper. These instances are motion planning in the plane amidst arbitrary obstacles, motion planning in 3-space amidst arbitrary obstacles of comparable sizes, and motion planning on a workfloor in 3-space amidst arbitrary obstacles. All three partitions are computable in nearly-optimal $O(n \log n)$ time. In conclusion, we have obtained $O(n \log n)$ motion planning algorithms for each of the three classes of problems, regardless of the number of degrees of freedom of the robot. The partitions that are obtained for the two remaining instances, motion planning in 3-space amidst polyhedral obstacles and amidst arbitrarily-shaped obstacles, have sizes $O(n^2)$ and $O(n^3)$ and are computable in time $O(n^2 \log n)$ and $O(n^3)$ respectively. Smaller workspace partitions may exist for three-dimensional instances involving polyhedral and arbitrary obstacles. An efficiently computable smaller decomposition will immediately result in an enhanced running time for the corresponding motion planning problem.

Besides attempting to improve the two non-optimal results, it is also interesting to see if the paradigm for motion planning in environments with low obstacle density applies to other classes of motion planning problems. The general idea of subdividing the configuration space into cylinders - with bases in some projective subspace - in which the free space has constant complexity may be applicable to configuration spaces other than $C = \mathrm{W} \times D$. The workfloor robot in Subsection 4.5 is an example of such a different type of problem. Other possible extensions include motion planning with moving obstacles, multiple robots, and anchored robot arms. We give some brief thoughts on these extensions, which we are currently investigating in more detail.

Motion planning problems involving moving obstacles are normally solved in the $(f+1)$-dimensional configuration-time space which is the Cartesian product of the f-dimensional configuration space C of the stationary version of the problem and the time dimension. When only a constant number c of obstacles move along algebraic curves of bounded degree, it seems sufficient to find a cc-partition of the workspace with the $n-c$ stationary obstacles. The number of moving obstacles and the nature of their trajectories guarantee that the constraint hypersurfaces induced by contacts of the robot and the moving obstacles do not increase the asymptotic complexity of the arrangement inside any cylinder obtained by lifting the cc-partition of the workspace. Therefore, the results of the previous sections seem to generalize directly to environments in which a constant number of the obstacles are non-stationary. Details though must be worked out.

The usual approach to the exact solution of a motion planning problem with c bounded-size robots with configuration spaces $C_1, \ldots, C_c$ of dimensions $f_1, \ldots, f_c$ is to regard these robots as one multi-body robot. Planning the motion of the multi-body robot takes place in the composite configuration space $C = C_1 \times \ldots \times C_c$. We believe the complexity of the free part of C to be close to the realizable lower bound of $\Omega(n^c)$ rather than to the trivial upper bound of $O(n^f)$, with $f = f_1 + \ldots + f_c$. The ideas of cylindrical decomposition of the free space seem applicable if the workspace $\mathrm{W} = \mathbb{R}^d$ is a projective subspace of each of the configuration spaces C_i. Then a point p in the projective subspace W^c of C fixes the positions of the reference points of all c bodies. The low obstacle density and the bounds on the sizes of the bodies yield that each body can touch only a constant number of obstacles while its reference point is fixed. Provided that c is a constant, the lifting of p into C is intersected by $O(1)$ constraint hypersurfaces. Hence, C is a cylindrifiable configuration space and W^c is a valid base space. The problem is to find a small base partition of the $(c \cdot d)$-dimensional space W^c.

In industrial robot arms, the links close to the hand - the minor axes - are generally considerably shorter than the links close to the base - the major axes. Consider an f-link robot arm of which the m links closest to the hand are not too large compared to the obstacles. Let C be the configuration space and assume that C' is the $(f-m)$-dimensional subspace corresponding to the major axes. A point $p \in C'$ fixes the placements of all major axes. If m is a constant, then the m minor axes can only touch a constant number of obstacles while the major axes are fixed, due to the low obstacle density. As a result, the lifting of the point $p \in C'$ into C will be intersected by only a constant number of constraint hypersurfaces, making C a cylindrifiable configuration space and C' a valid base space. It is however unclear for the moment how to compute base partitions in C'.

References

[1] P.K. Agarwal, M.J. Katz, and M. Sharir, Computing depth orders and related problems, *Proc. 4th Scandinavian Workshop on Algorithm Theory (SWAT'94)* (E.M. Schmidt and S. Skyum Eds.) Lecture Notes in Computer Science **824** (1994), pp. 1-12.

[2] H. Alt, R. Fleischer, M. Kaufmann, K. Mehlhorn, S. Näher, S. Schirra, and C. Uhrig, Approximate motion planning and the complexity of the boundary of the union of simple geometric figures, *Algorithmica* **8** (1992), pp. 391-406.

[3] B. Aronov and M. Sharir, Triangles in space or building (and analyzing) castles in the air, *Combinatorica* **10** (1990), pp. 137-173.

[4] F. Avnaim, J.-D. Boissonnat, and B. Faverjon, A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles, *Proc. Geometry and Robotics Workshop* (J.-D. Boissonnat and J.-P. Laumond Eds.), Lecture Notes in Computer Science **391** (1988), pp. 67-86.

[5] J.L. Bentley and T.A. Ottman, Algorithms for reporting and counting geometric intersections, *IEEE Transactions on Computers* **28** (1979), pp. 643-647.

[6] M. de Berg, M. de Groot, and M. Overmars, New results on binary space partitions in the plane, *Proc. 4th Scandinavian Workshop on Algorithm Theory (SWAT'94)* (E.M. Schmidt and S. Skyum Eds.) Lecture Notes in Computer Science **824** (1994), pp. 61-72.

[7] M. de Berg, L.J. Guibas, and D. Halperin, Vertical decompositions for triangles in 3-space, *Proc. 10th Ann. ACM Symp. on Computational Geometry* (1994), pp. 1-10.

[8] J.F. Canny, *The complexity of robot motion planning*, MIT Press, Cambridge MA (1988).

[9] B. Chazelle, Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm, *SIAM Journal on Computing* **13** (1984), pp. 488-507.

[10] A. Efrat, G. Rote, and M. Sharir, On the union of fat wedges and separating a collection of segments by a line, *Computational Geometry: Theory and Applications* **3** (1993), pp. 277-288.

[11] L. Guibas and J. Stolfi, Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams, *ACM Transactions on Graphics* **4** (1985), pp. 74-123.

[12] D. Halperin and M.H. Overmars, Spheres, molecules, and hidden surface removal, *Proc. 10th Ann. ACM Symp. on Computational Geometry* (1994), pp. 113-122.

[13] J. Hershberger, Finding the upper envelope of n line segments in $O(n \log n)$ time, *Information Processing Letters* **33** (1989), pp. 169-174.

[14] M.J. Katz, M.H. Overmars, and M. Sharir, Efficient hidden surface removal for objects with small union size, *Computational Geometry: Theory and Applications* **2** (1992), pp. 223-234.

[15] Y. Ke and J. O'Rourke, Moving a ladder in three dimensions: upper and lower bounds, *Proc. 3rd Ann. ACM Symp. on Computational Geometry* (1987), pp. 136-145.

[16] M. van Kreveld, On fat partitioning, fat covering and the union size of polygons, Technical Report RUU-CS-93-36, Dept. of Computer Science, Utrecht University (1993).

[17] D. Leven and M. Sharir, Planning a purely translational motion for a convex object in two-dimensional space using generalized Voronoi diagrams, *Discrete & Computational Geometry* **2** (1987), pp. 9-31.

[18] D. Leven and M. Sharir, An efficient and simple motion planning algorithm for a ladder amidst polygonal barriers, *Journal of Algorithms* **8** (1987), pp. 192-215.

[19] J. Matoušek, J. Pach, M. Sharir, S. Sifrony, and E. Welzl, Fat triangles determine linearly many holes, *SIAM Journal on Computing* **23** (1994), pp. 154-169.

[20] C. Ó'Dúnlaing, M. Sharir, and C.-K. Yap, Retraction: A new approach to motion planning, *Proc. 15th Ann. ACM Symp. on the Theory of Computing* (1983), pp. 207-220.

[21] C. Ó'Dúnlaing and C.-K. Yap, A retraction method for planning the motion of a disc, *Journal of Algorithms* **6** (1985), pp. 104-111.

[22] M.H. Overmars, Point location in fat subdivisions, *Information Processing Letters* **44** (1992), pp. 261-265.

[23] M.H. Overmars and A.F. van der Stappen, Range searching and point location among fat objects, *Journal of Algorithms*, to appear. (Also available as: Tech. Rep. UU-CS-1994-30, Dept. of Computer Science, Utrecht University (1994).)

[24] Ph. Pignon, *Structuration de l' espace pour une planification hiérarchisée des trajectoires de robots mobiles*, Ph.D. Thesis, LAAS-CNRS and Université Paul Sabatier de Toulouse, Rapport LAAS N^o 93395 (1993) (in French).

[25] J.T. Schwartz and M. Sharir, On the piano movers' problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal boundaries, *Communications on Pure and Applied Mathematics* **36** (1983), pp. 345-398.

[26] J.T. Schwartz and M. Sharir, On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds, *Advances in Applied Mathematics* **4** (1983), pp. 298-351.

[27] J.T. Schwartz and M. Sharir, On the piano movers' problem: III. Coordinating the motion of several independent bodies: the special case of circular bodies moving amidst polygonal barriers, *International Journal of Robotics Research* **2** (1983), pp. 46-75.

[28] J.T. Schwartz and M. Sharir, On the piano movers' problem: V. The case of a rod moving in three-dimensional space amidst polyhedral obstacles, *Communications on Pure and Applied Mathematics* **37** (1984), pp. 815-848.

[29] J.T. Schwartz and M. Sharir, Efficient motion planning algorithms in environments of bounded local complexity, Report 164, Department of Computer Science, Courant Inst. Math. Sci., New York NY (1985).

[30] M. Sharir, Efficient algorithms for planning purely translational collision-free motion in two and three dimensions, *Proc. of the IEEE Int. Conf. on Robotics and Automation*, Raleigh NC (1987), pp. 1326-1331.

[31] M. Sharir and E. Ariel-Sheffi, On the piano movers' problem: IV. Various decomposable two-dimensional motion planning problems, *Communications on Pure and Applied Mathematics* **37** (1984), pp. 479-493.

[32] S. Sifrony and M. Sharir, A new efficient motion planning algorithm for a rod in two-dimensional polygonal space, *Algorithmica* **2** (1987), pp. 367-402.

[33] A.F. van der Stappen, D. Halperin, M.H. Overmars, The complexity of the free space for a robot moving amidst fat obstacles, *Computational Geometry: Theory and Applications* **3** (1993), pp. 353-373.

[34] A.F. van der Stappen, The complexity of the free space for motion planning amidst fat obstacles, *Journal of Intelligent and Robotic Systems* **11** (1994), pp. 21-44.

[35] A.F. van der Stappen and M.H. Overmars, Motion planning amidst fat obstacles, *Proc. 10th Ann. ACM Symp. on Computational Geometry* (1994), pp. 31-40.

[36] A.F. van der Stappen, *Motion planning amidst fat obstacles*, Ph.D. Thesis, Dept. of Computer Science, Utrecht University (1994).

[37] C. Wentink and O. Schwarzkopf, Motion planning for vacuum cleaner robots, *Proc. 6th Canadian Conf. on Computational Geometry* (1994), pp. 51-56.

[38] A. Wiernik and M. Sharir, Planar realizations of nonlinear Davenport-Schinzel sequences by segments, *Discrete & Computational Geometry* **3** (1988), pp. 15-47.