Motion Planning in Environments with Low Obstacle Density *†

A. Frank van der Stappen Mark H. Overmars Mark de Berg Jules Vleugels

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 leads to nearly-optimal $O(n \log n)$ motion planning algorithms for free-flying robots with any fixed number of degrees of freedom in workspaces with low obstacle density.

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 [22, 23, 24, 25, 28]—by Schwartz and Sharir in the early 80s, studies the exact solution of the motion planning problem. Exact methods for solving the 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,

*Research is supported by the Dutch Organization for Scientific Research (N.W.O.) and by the ESPRIT III BRA Project 6546 (PROMotion).

†Revision of Technical Report UU-CS-1995-33, Dept. of Computer Science, Utrecht University (1995).

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 free 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 semi-algebraic 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 $\mathcal{E}$:

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

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. [12, 15, 22, 23, 24, 25, 28]) 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. [7, 14, 17, 18, 29]) 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 $Z \in \mathrm{FP}$ to some point $\mathrm{Im}(Z)$ 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 is reduced to the problem of finding a sequence of roadmap curves connecting the roadmap points $\mathrm{Im}(Z_0)$ and $\mathrm{Im}(Z_1)$. 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 later on 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 obtained for specific free-flying

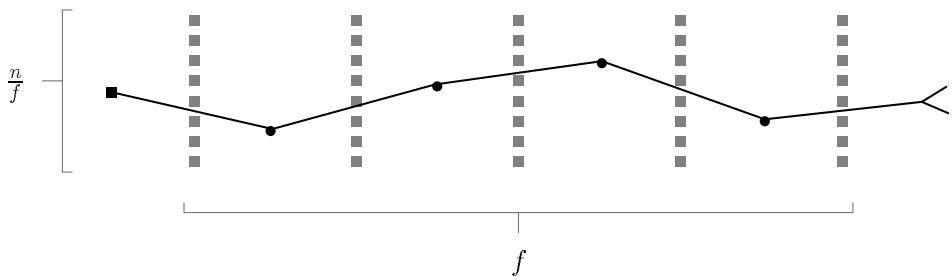

Figure 1: An (f-DOF) robot arm consisting of f links with $\Omega(n^f)$ f-fold contacts, and, hence, with free space complexity $\Omega(n^f)$.

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. [27, 36]). 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 cannot 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 impressive gap between the $\Omega(n^f)$ construction and

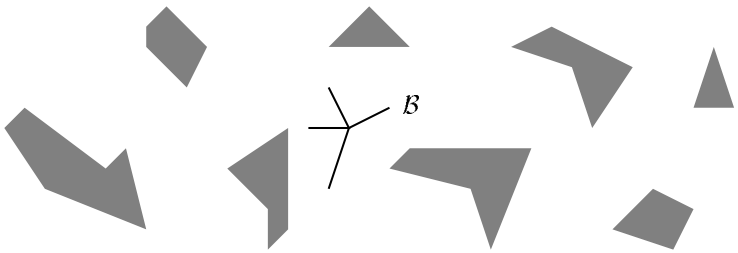

Figure 2: A 6-DOF robot with few multiple contacts, and, hence, with low free space complexity.

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 [30] 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 intersects no more than a constant number of obstacles that are at least as large as the region. We shall refer to the latter property as the *low obstacle density* property of the workspace. (Actually, in [30] 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 [26] who refer to it as *bounded local complexity* and by Pignon [21] who calls it *sparsity*.

A question that immediately comes to mind when considering the combinatorial result of [30] 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 [29] 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 could be $\Theta(n^2)$ in the worst case for arbitrary workspaces with obstacles. Some algorithms have a hidden sensitivity to the complexity of the free space [33]. For example, the boundary cell decomposition algorithm by Avnaim, Boissonnat, and Faverjon [3], 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 [22] 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 improves the efficiency to a running time of $O(n \log n)$ (see also [31]). 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 [23] with running time $O(n^{2^{f+6}})$ and Canny [7] 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 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 asymptotic complexity of the decomposition. The realistic low obstacle density and bounded robot size assumptions also guarantee the existence of a workspace partition of (optimal) $O(n)$ size, based on the binary space partition by De Berg [4]. The computation of the partition takes $O(n \log n)$ time. As a result, motion planning problems in low obstacle density environments can be solved in $O(n \log n)$ time. Notice that the bounds do 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 [29] 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 [26] 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 [21] structures two-dimensional workspaces with polygonal obstacles and a polygonal robot (using Minkowski differences) to efficiently detect and solve simple path-finding queries. Simple queries are queries that are either easily seen to yield no solution—because there exists no path for a simple inscribed shape of the robot—or easily solvable —because there exists a path for an outer approximation of the robot with fewer degrees of freedom. 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 properties of 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 an algorithm based on the paradigm depends on the time to compute some constrained

partition of the workspace. Section 4 tackles the problem of finding a small and efficiently computable partition. 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 [30], 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 a reference point on the robot, 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 $b \geq 0$. Definition 2.2 (see also [34]) 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.

Definition 2.2 [low (object/obstacle) density]
Let $\mathbb{R}^d$ be a space with a set $\mathcal{E}$ of objects. Then $\mathbb{R}^d$ is said to be a low (object) density space *if any region with minimal enclosing hypersphere radius σ intersects at most a constant number of objects $E \in \mathcal{E}$ with minimal enclosing hypersphere radius at least σ.*

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.

Lemma 2.3 follows easily if we realize that we can cover a region with minimal enclosing hypersphere radius $c \cdot \sigma$ by $(\lfloor 2c \rfloor + 1)^d$ hyperspheres with radius σ, each of which intersects no more than a constant number of objects.

Lemma 2.3 *Let $\mathbb{R}^d$ with a set $\mathcal{E}$ of objects satisfy the low object density property. Then any region with minimal enclosing hypersphere radius $c \cdot \sigma$, for some constant $c \geq 0$, intersects at most a constant number of objects $E \in \mathcal{E}$ with minimal enclosing hypersphere radius at least σ.*

Another immediate consequence of the low object density property is that every point $p \in \mathbb{R}^d$ lies in at most a constant number of objects $E \in \mathcal{E}$. This fact, in conjunction with Theorem 2.4 *and* the invariance of the low density property under moderate inflation of the objects (expressed by Theorem 2.9), is crucial in the considerations of Section 3.

Theorem 2.4 *Let $\mathbb{R}^d$ with a set $\mathcal{E}$ of n constant-complexity objects satisfy the low object density property. Then the complexity of the arrangement*[1] *$\mathcal{A}(\partial\mathcal{E})$ of all object boundaries ∂E is $O(n)$.*

Proof: Let us assume that the objects in $\mathcal{E}$ are ordered by increasing minimal enclosing hypersphere radius: $E_1, \ldots, E_n$ and $\rho_1 \leq \ldots \leq \rho_n$, where ρ_i is the minimal enclosing hypersphere radius of E_i.

We count for each object boundary ∂E_i the subspaces of dimensions 0 through $d-1$ that are defined by its intersection with object boundaries ∂E_j with $j > i$. A boundary ∂E_i can only be intersected by a boundary ∂E_j $(j > i)$ if the minimal enclosing hypersphere S_i (with diameter ρ_i) of E_i is intersected by E_j. Definition 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 boundaries ∂E_j $(j > i)$ that intersect ∂E_i. By the additional assumption that all objects and thus their boundaries 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 ∂E_i and boundaries ∂E_j $(j > i)$. Adding the contributions of all n boundaries amounts to a total of $O(n)$ subspaces of dimensions 0 to $d-1$ in the arrangement $\mathcal{A}(\partial\mathcal{E})$. The linear bounds on the number of these subspaces imply the same bound of $O(n)$ on the number of d-faces in $\mathcal{A}(\partial\mathcal{E})$, making the total combinatorial complexity of the arrangement $O(n)$. □

Theorem 2.5 states the linear complexity result for the free space for motion planning problems in low density environments. The reader is referred to [30] for a proof.

[1] The arrangement of a set is the subdivision of space into connected pieces of any dimension induced by that set.

Theorem 2.5 *Let* W *with a set* $\mathcal{E}$ *of* n *constant-complexity obstacles with minimal enclosing hypersphere radii at least* ρ *be a low obstacle density workspace. The free space for a constant-complexity robot* $\mathcal{B}$ *with reach* $\rho_{\mathcal{B}} \leq b \cdot \rho$*, for some constant* $b \geq 0$*, moving in* W *has complexity* $O(n)$*.*

In the next subsection, we consider an interesting class of motion planning environments that satisfy the low obstacle density property. An immediate consequence of the preceding results will be that complexity of the free space is linear in the number of obstacles.

2.2 Fatness

Fatness has turned out to be an interesting phenomenon in computational geometry. Several papers present surprising improvements in combinatorial complexity bounds [2, 9, 13, 16, 30] and efficiency gains for algorithms [1, 4, 11, 19, 20] if the objects under consideration are fat. 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 improved 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, 9, 13, 16], a linear bound on the complexity of the free space for motion planning amidst fat obstacles [30], and efficient algorithms for computing depth orders on certain fat objects [1], binary space partitions for sets of fat objects [4], hidden surface removal for fat horizontal triangles [11], and range searching and point location among fat objects [19, 20].

Contrary to many other definitions of fatness in literature [1, 2, 9, 11, 13, 16], the notion introduced in [30], 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 2.6 [k**-fatness**]
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 does not fully contain 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 obvious at first sight. An intuitive explanation lies in the observation that it is impossible to have a large number of fat obstacles of a certain minimum size intersecting a small region. A more formal proof follows from [30]. Here, we confine ourselves to reporting the result.

Theorem 2.7 *A space* $\mathbb{R}^d$ *with non-intersecting* k*-fat objects is a low object density space.*

2.3 Object wrappings

This subsection shows that the objects in a scene satisfying the low density property can be expanded by an amount proportional to the size of the smallest object without affecting the

low density property. Intuitively, the first objects that are to be intersected by expanding obstacles are 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 density of the scene. Below, these informal ideas are made specific by giving bounds on the (allowable) expansion of the objects such that the density of the space with the expanded objects remains $O(n)$. The 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, these wrappings also help in finding an efficient instance of the paradigm for the large class of problems involving a free-flying robot.

Definition 2.8 [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 2.9 states the circumstances that preserve the low density property of the space while the objects are expanded. 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 2.9 *Let $\mathbb{R}^d$ with a set $\mathcal{E}$ of objects satisfy the low object density property, and let ρ be a lower bound on the minimal enclosing hyperspere radii of the objects in $\mathcal{E}$. Let $b \geq 0$ be a constant and assume that a $(b \cdot \rho)$-wrapping $\Delta(E)$ is given for every object $E \in \mathcal{E}$. Then $\mathbb{R}^d$ with the set $\{\Delta(E) | E \in \mathcal{E}\}$ of wrappings satisfies the low object density property.*

Proof: Any region in $\mathbb{R}^d$ with a minimal enclosing hypersphere radius σ intersects at most a constant number of objects from $\mathcal{E}$ with a minimal enclosing hypersphere radius at least σ, because $\mathbb{R}^d$ with the objects $\mathcal{E}$ satisfies the low object density property. Let $R \subseteq \mathbb{R}^d$ be a region with a minimal enclosing hypersphere radius σ. Let S be the hypersphere with a radius $\sigma + b\rho$ and concentric with R's minimal enclosing hypersphere. We count the number of wrappings with a minimal enclosing hypersphere radius at least σ that intersect R. Let $\Delta(E)$ be such a wrapping. We note that if $\Delta(E)$ intersects R, then the object E itself must intersect the interior of the hypersphere S with radius $\sigma + b\rho$. Moreover, we note that the minimal enclosing hypersphere radius of E itself is at least $\sigma - b\rho$. Thus, we can bound the number of wrappings with minimal enclosing hypersphere radius at least σ intersecting R by bounding the number of objects with minimal enclosing hypersphere radius at least $\sigma - b\rho$ intersecting S. We distinguish two cases.

Assume $\sigma \geq 2b\rho$. Then, the radius $\sigma + b\rho$ of S satisfies the inequality $\sigma + b\rho \leq 3(\sigma - b\rho)$. By Lemma 2.3, S intersects at most a constant number of objects with minimal enclosing hypersphere radius at least $\sigma - b\rho$.

Assume $\sigma \leq 2b\rho$. Then, the radius $\sigma + b\rho$ of S satsifies $\sigma + b\rho \leq 3b\rho$. By Lemma 2.3, S is intersected by at most a constant number of objects. □

Informally, Theorem 2.9 says that the low object density property is preserved if the objects in a low density space are expanded by an amount that is at most proportional to the size of the

smallest object in the scene. We shall use Theorem 2.9 mostly in conjunction with Theorem 2.4; together these theorems imply that the arrangement of expanded obstacle boundaries has linear complexity, given that each expanded obstacle has constant complexity.

Besides applications in motion planning, Theorems 2.4 and 2.9 have interesting implications for certain types of arrangements and 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 2.4, the boundary of the union of all wrappings has complexity $O(n)$. Theorems 2.4 and 2.9 are, for example, applicable to the molecule model in the paper by Halperin and Overmars [10]. 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 get too close to another sphere center. This property allows to regard the atoms as wrappings of certain non-intersecting smaller spheres, which are only a bounded amount smaller than the original atoms and can be shown to satisfy the low density property. The construction provides an alternative proof for the linear (in the number of atoms) descriptional complexity of the molecule surface.

3 A paradigm for low density motion planning

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 [23] and Canny [7]) 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 and forbidden cells. Even in the case of low density motion planning, the complexity of a single free cell can be $\Omega(n)$, 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 capture 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

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

for each point $p \in \mathrm{W}$. 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

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

for each point $p \in B$. 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.

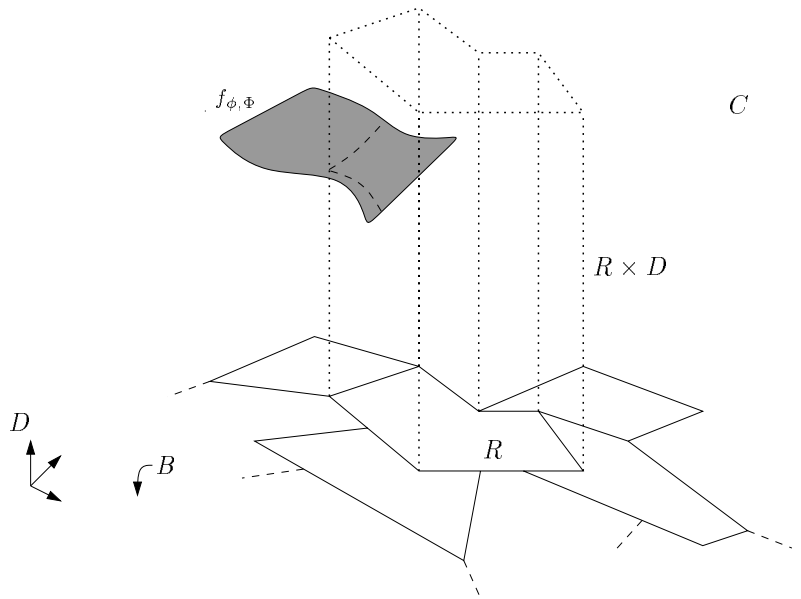

Figure 3: A three-dimensional example of a cylindrifiable configuration space C with a two-dimensional base space B (and, hence, a one-dimensional rest-space D), and a fragment of the base partition in B. The constraints on the base partition guarantee that the cylinder $R \times D$ is intersected by 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 $R \times D$ 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 motion planning for free-flying robots.

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 $E_B = \{(R, R') \in V_B \times V_B | \partial R \cap \partial R' \neq \emptyset\}$ contains the adjacencies of V_B's regions. 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: $|V_C| = O(|V_B|)$ and $|E_C| = O(|E_B|)$. 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.

Algorithm 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 FP $\cap (R \times D)$ into
   a constant number of constant-complexity subcells A;
3. $Desc(R) := \emptyset$;
4. **for all** subcells A of 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. We review the different steps of

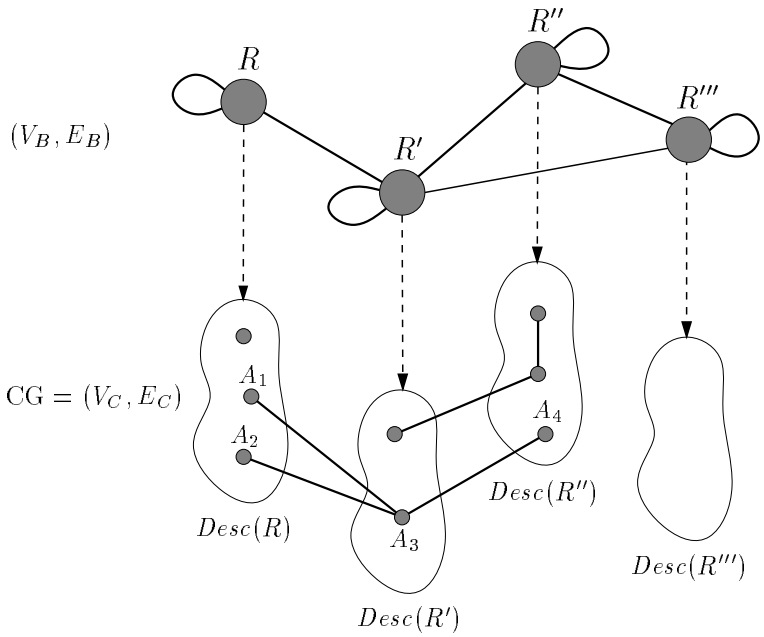

Figure 4: The relation between the base partition graph (V_B, E_B) in the subspace B of C at the top, and the connectivity graph $\mathrm{CG} = (V_C, E_C)$ in the configuration space at the bottom. Each node $R \in V_B$ defines at most $O(1)$ nodes $A \in V_C$, collected in a set $Desc(R)$. Two nodes A and A' in V_C can only be connected if the corresponding nodes R and R' in V_B are connected, so e.g. A_1 may be connected to nodes in $Desc(R')$ but not to nodes in $Desc(R)$.

the transformation in more detail to verify their validity and to analyze 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 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 [23] to the constant number of constraint hypersurfaces intersecting the cylinder $R \times D$. The output 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 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. If this requirement is indeed satisfied, 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 in $A_1 \in Desc(R_1)$ and $A_2 \in Desc(R_2)$ 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* 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 the property of Definition 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 a 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) [35]. 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 the two-dimensional subspace of the workspace of all possible positions of the reference point.

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. Lemma 2.3 says 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 $|\{f_{\phi,\Phi} | \phi \in_f \mathcal{B} \wedge \Phi \in_f \mathcal{E} \wedge f_{\phi,\Phi} \cap (p \times D) \neq \emptyset\}| = O(1)$, *for all* $p \in \mathrm{W}$.

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 $p \times D$. □

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:

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

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.[2] Clearly, the robot's reference point must lie inside $G(E, \rho_{\mathcal{B}})$ in order for a feature ϕ of the robot $\mathcal{B}$ to be in contact with a feature Φ of the obstacle E; if the reference point lies outside $G(E, \rho_{\mathcal{B}})$ there is no danger for $\phi \in_f \mathcal{B}$ of colliding with $\Phi \in_f E$. A formalization of these observations leads to an interesting property on the 'location' of a constraint hypersurface in configuration space.

Lemma 3.4 *$f_{\phi,\Phi} \subseteq G(E, \rho_{\mathcal{B}}) \times D$, for all $\phi \in_f \mathcal{B}$ and $\Phi \in_f E$.*

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 $\Phi \in_f E$ may 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 $|Cov(R)| = O(1)$.

[2] $A \oplus B = \{a + b | a \in A \wedge b \in B\}$.

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 since the space $\mathbb{R}^d$ with the grown obstacles $G(E, \rho_{\mathcal{B}})$ is a low obstacle density space. The fact follows immediately from Theorem 2.9, noting that each grown obstacle $G(E, \rho_{\mathcal{B}})$ is a $\rho_{\mathcal{B}}$-wrapping and, by $\rho_{\mathcal{B}} \leq b \cdot \rho$, also a $(b \cdot \rho)$-wrapping of the obstacle E itself. As a result, it is indeed possible to partition the low obstacle density workspace W into regions with constant-size coverage.

Lemma 3.8 *A low obstacle density workspace* W *can be partitioned into regions* R *with* $|Cov(R)| = O(1)$.

Notice that, by Theorem 2.4, the arrangement $\mathcal{A}(\partial G)$ of grown obstacle boundaries $\partial G(E, \rho_{\mathcal{B}})$ 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. 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. In the case of a planar workspace, the partition is easily refined into a cc-partition of $O(n)$ size by means of a vertical decomposition [32, 33]. The decomposition procedure discussed in Section 4, however, gives a cc-partition of linear size for any dimension.

In summary, we have found that a cc-partition of a low obstacle density workspace always exists. The cc-partition of 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 LODMOT 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 $Cov : V_{\mathrm{W}} \to \mathcal{P}(\mathcal{E})$ 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 LODMOT

Find a cc-partition graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and compute $Cov : V_{\mathrm{W}} \to \mathcal{P}(\mathcal{E})$;
$(V_C, E_C) :=$ TRANSFORM$((V_{\mathrm{W}}, E_{\mathrm{W}}))$

The $|V_{\mathrm{W}}|$ 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 reveals 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 [23] 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 $O(|V_{\mathrm{W}}|)$ for the first **for**-loop of the transformation. The running time of the entire algorithm LODMOT becomes $O(|V_{\mathrm{W}}| + |E_{\mathrm{W}}| + T(n))$, by Lemma 3.1 and the assumption that the computation of the cc-partition and function Cov takes $T(n)$ time. The $O(|V_{\mathrm{W}}| + |E_{\mathrm{W}}| + T(n))$ time bound emphasizes once again that the efficiency of LODMOT is fully determined by the size of the graph $(V_{\mathrm{W}}, E_{\mathrm{W}})$ and the time to compute it along with $Cov : V_{\mathrm{W}} \to \mathcal{P}(\mathcal{E})$. Since the time $T(n)$ to compute the graph and the function dominates the time $O(|V_{\mathrm{W}}| + |E_{\mathrm{W}}|)$ to just report both, we may conclude that the $T(n)$-factor dominates the running time of the algorithm LODMOT, which may therefore be said to equal $O(T(n))$.

Theorem 3.9 *Let* W *with a set* $\mathcal{E}$ *of obstacles with minimal enclosing hypersphere radii at least* ρ *be a low obstacle density workspace. Let* $\mathcal{B}$ *be a constant-complexity robot with reach* $\rho_{\mathcal{B}} \leq b \cdot \rho$*, for some constant* $b \geq 0$*, moving in* W*. Furthermore, let* $C = \mathrm{W} \times D$ *be the configuration space of* $\mathcal{B}$*. Then, the algorithm* LODMOT *computes the connectivity graph* (V_C, E_C) *with* $|V_C| = O(|V_{\mathrm{W}}|)$ *and* $|E_C| = O(|E_{\mathrm{W}}|)$ *of a cell decomposition of the free space* $\mathrm{FP} \subseteq C$ *in time* $O(T(n))$*, where* $(V_{\mathrm{W}}, E_{\mathrm{W}})$ *is a cc-partition of* W *and* $T(n)$ *is the time to compute* $(V_{\mathrm{W}}, E_{\mathrm{W}})$ *along with* $Cov : V_{\mathrm{W}} \to \mathcal{P}(\mathcal{E})$*.*

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. The next section is devoted to providing small and efficiently computable cc-partitions, 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 [33] for further details.

4 A linear size base partition

In Subsection 3.2, we have reduced the low density motion planning problem for a free-flying robot to the problem of finding a cc-partition of the workspace. A cc-partition subdivides the robot's workspace W into constant-complexity regions that intersect no more than a constant number of grown obstacles. The complexity of this partition, i.e., the number of regions and region adjacencies, and the time to compute it determine the running time of the resulting motion planning algorithm, so we must try to find a small and efficiently computable cc-partition. This section discusses a cc-partition of linear size, which can be computed in nearly linear time.

We obtain a cc-partition of the workspace by applying a simplified version of the first stage of De Berg's binary space partition algorithm [4] to the set $\{G(E, \rho_{\mathcal{B}}) | E \in \mathcal{E}\}$ of grown obstacles. The procedure takes the set of vertices of the axis-parallel bounding boxes of the grown obstacles as input and outputs a decomposition of the workspace into a linear number of hypercubic and L-shaped regions without bounding box vertices in their interiors. It turns out that each of these regions intersects only a constant number of grown obstacles. Below, we explain the procedure in detail.

Let S be a set of n objects and let Σ be the set of vertices of the axis-parallel bounding boxes of these objects. The set Σ contains $2^d n$ vertices. For a region R, we denote by $\Sigma(R)$ the subset of Σ of bounding box vertices contained in the interior of R. Assume that all bounding boxes are enclosed by a large axis-parallel hypercube, which contains all vertices of Σ. Our variant of the first stage of the binary space partition algorithm by De Berg [4] recursively subdivides this large hypercube until all resulting regions R satisfy $\Sigma(R) = \emptyset$. Let C be a hypercube with $\Sigma(C) \neq \emptyset$ and let $C_1, \ldots, C_{2^d}$ be the equally-sized sub-hypercubes resulting from a 2^d-tree split of C, i.e., from cutting C with the d hyperplanes perpendicular to and bisecting its edges. Note that every sub-hypercube C_i shares exactly one corner with C. A 2^d-tree split is called *useless* if all vertices of $\Sigma(C)$ lie in a single sub-hypercube C_i of C, and called *useful* otherwise. The hypercube C is subdivided in one of the following ways.

- If the 2^d-tree split of C is useful, then perform the 2^d-tree split.
- If the 2^d-tree split of C is useless, then replace the single sub-hypercube C_i containing all vertices of $\Sigma(C)$ by the smallest hypercube that shares a corner with C and still contains all vertices of $\Sigma(C)$. The resulting hypercube $C' \subseteq C$ has one of the vertices of $\Sigma(C)$ on its boundary. Although De Berg refines the complement $C \setminus C'$ into d boxes in order to obtain a binary space partition, we simply split C into the smaller hypercube C' and the 'L-shaped' complement $C \setminus C'$, which satisfies $\Sigma(C \setminus C') \neq \emptyset$. We refer to this type of split of C as an L-split.

Figure 5 gives two-dimensional examples of both types of splits. The recursive splits lead to a decomposition into $O(n)$ hypercubic and L-shaped regions that have no vertices from Σ in their interiors. The computation of the tree corresponding to the recursive decomposition—along with the sets of objects intersecting each of the regions—takes $O(n \log n)$ time. The nodes of the tree can be rearranged in $O(n \log n)$ time into a second tree of height $O(\log n)$ (see [4] for details). This second tree allows us to do point location in the decomposition in $O(\log n)$ time, provided that it is possible to determine in constant time for every internal node v whether a query point lies in one of regions associated with the children of v, or in the complement of the region associated with v. The condition is true for our decomposition,

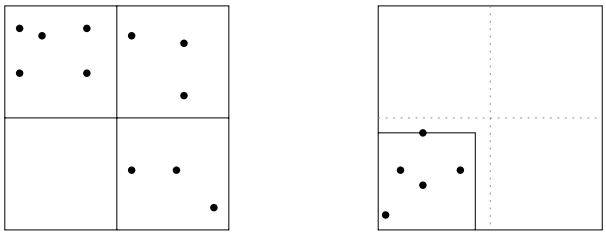

Figure 5: A useful and useless 2^d-tree split. In the latter case the square is subdivided into an L-shaped region and a square.

because its regions are either hypercubes or L-shapes. We will use the point location structure during the computation of the region adjacencies.

De Berg [4] shows that, under the weak condition of *unclutteredness*, the number of objects from S intersecting each of the boxes of his decomposition is constant. Since our decomposition only differs from De Berg's first stage decomposition in that our L-shaped regions are the union of a constant number of his boxes (see above), this property is valid for our decomposition as well. Definition 4.1 recalls the unclutteredness condition.

Definition 4.1 uncluttered
Let $\mathbb{R}^d$ be a space with a set S of objects. The set S satisfies the uncluttered condition *if there is a constant κ such that any hypercube whose interior does not contain a vertex of one of the bounding boxes of S is intersected by at most κ objects of S.*

In [5], De Berg et al. study the relations between various realistic assumptions on sets of geometric objects, including fatness, unclutteredness, and low density. Among their results is the following relation.

Lemma 4.2 [5] *Let $\mathbb{R}^d$ with a set S of objects satisfy the low object density property. Then $\mathbb{R}^d$ with S satisfies the uncluttered condition.*

Let us now apply the recursive decomposition scheme to the set $\{G(E, \rho_{\mathcal{B}}) | E \in \mathcal{E}\}$. As all grown obstacles are $(b \cdot \rho)$-wrappings of the original obstacles, and the original obstacles have minimal enclosing hypersphere radius at least ρ, we know by Theorem 2.9 that the workspace with the set $\{G(E, \rho_{\mathcal{B}}) | E \in \mathcal{E}\}$ of grown obstacles is still a low density space. Thus, if we apply the decomposition to $\{G(E, \rho_{\mathcal{B}}) | E \in \mathcal{E}\}$, then Lemma 4.2 assures that we end up with a partition V_{W} of the workspace into regions R that each intersect at most a constant number of grown obstacles, or, in other words, with $|Cov(R)| = O(1)$. As the boxes clearly have constant complexity in a fixed dimension d, we find that the partition V_{W} is a cc-partition. The set V_{W} has size $O(n)$ and is computable, along with the mapping $Cov : V_{\mathrm{W}} \to \mathcal{P}(\mathcal{E})$, in $O(n \log n)$ time.

It remains to bound the number of pairwise adjacencies of regions of V_{W}, and to show how to efficiently compute the set of adjacencies. It turns out to be convenient (at certain occasions) to cover the L-shaped regions in our decomposition by imaginary hypercubes. In an L-split, a hypercube C is subdivided into a small hypercube C' and an L-shaped complement $C \setminus C'$. Let s and s' be the side lengths of the hypercubes C and C' respectively. We cover $C \setminus C'$ by $2^d - 1$ (which is a constant in a fixed dimension d) imaginary hypercubes of side length

$s-s'$. This is achieved as follows: for each of the 2^d-1 corners of C not occupied by C', we take an imaginary hypercube that lies entirely inside C and has one of its corners coinciding with that corner of C. Figure 6 shows the three squares that cover a two-dimensional L-shaped region. Each L-shaped region can thus be covered by $2^d - 1$ imaginary hypercubes. The

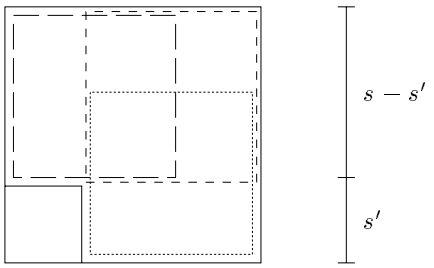

Figure 6: Three imaginary squares—shown slightly smaller than their actual size for reasons of clarity—cover the L-shaped region.

$O(n)$ hypercubes from the decomposition and the $O(n)$ imaginary hypercubes jointly cover the entire workspace. A useful observation is that if two hypercubes are adjacent, then one of them has a corner in the interior of a facet of the other hypercube, unless both hypercubes are of equal size and share an entire facet. In this specific case, however, the center of one these facets, i.e., the point with equal distance to all facet boundaries, lies in the interior of the other facet (and vice versa). In summary, if we define the set of characteristic points of a hypercube to consist of its 2^d corners and its $2d$ facet centers, then for any two adjacent hypercubes, one of them has a characteristic point in the interior of a facet of the other.

Lemma 4.3 *The set E_W of pairwise adjacencies of regions in V_W has size $O(n)$.*

Proof: If we replace each L-shaped region by the $2^d - 1$ hypercubes, the number of pairwise adjacencies of the hypercubes in the newly obtained covering of the workspace is larger than the number of pairwise adjacencies of regions of the original decomposition. Hence, we can bound the size of E_W by bounding the number of adjacencies in the covering.

We first show how to charge every pairwise adjacency to a characteristic point on a hypercube and then bound the number of times a point gets charged by a constant. Recall that one of two adjacent hypercubes has a characteristic point in the interior of a facet of the other hypercube. We charge the adjacency to this point. As a region of the decomposition is covered by at most $2^d - 1$ hypercubes simultaneously, no characterstic point can lie on the boundary of more than $2^d - 1$ hypercubes. A characteristic point can therefore lie on the common boundary of at most a constant number of pairs of hypercubes. In other words, each of the $O(n)$ characteristic points is involved in no more than a constant number of hypercube adjacencies. The combination with the fact that every adjacency can be charged to a characteristic point yields the desired result. □

We compute the set E_W of adjacencies by a reverse execution of the subdivision process: starting from the hypercubic and L-shaped regions of the final decomposition, we repeatedly join the regions resulting from a single split until we obtain the initial hypercube containing

all grown obstacles and their bounding boxes. While joining the regions resulting from a split, we compute all adjacencies of regions from V_{W} that are created by doing so. For each of the $2d$ facets f of a hypercube C involved in a joining step, we take care to have available the set $\gamma(f)$ of sub-regions of C (from V_{W}) that share a part of their boundary with f.

First, let us see what happens if we join an L-shaped region L and a hypercube C. Recall that L is a region of the final decomposition V_{W}. The region L becomes adjacent to all regions in C that share a facet with the d facets of C that are 'glued' onto facets of L (see Figure 7 for a 3D example). In other words, for each such facet f of C, the region L becomes adjacent to all regions in $\gamma(f)$. The time required to report all different adjacencies of L is clearly

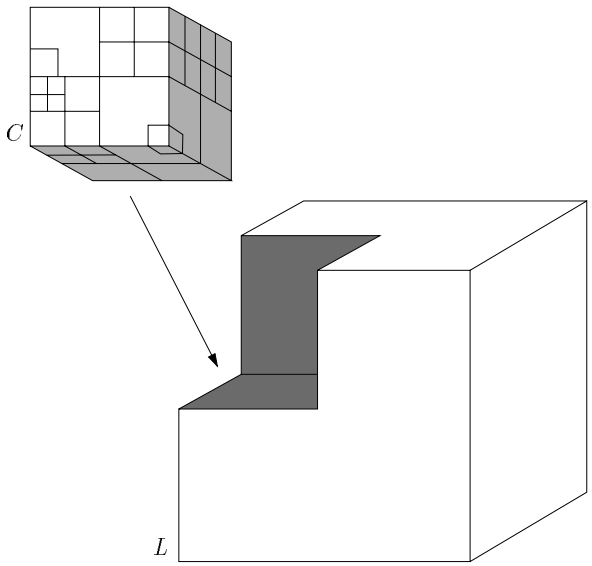

Figure 7: The grey facets of C are glued onto facets of L: L becomes adjacent to all regions in C that share a part of their boundary with the grey facets.

proportional to the number of such adjacencies. The sets of regions stored with the facets of the hypercubic union of L and C are easily computed from L and the sets stored with the facets of C.

Second, let us see what happens if we join 2^d equally-sized hypercubes into one larger hypercube. The basic task, which has to be performed $d2^{d-1}$ times, is to glue a $(d-1)$-dimensional facet f of a hypercube C onto a $(d-1)$-dimensional facet f' of another hypercube C' (see Figure 8 for a 3D example). This will cause pairs of regions from C and C', or more specifically, pairs of regions from $\gamma(f)$ and $\gamma(f')$, to become adjacent. To compute the adjacencies, we replace each L-shaped region in $\gamma(f)$ and $\gamma(f')$ by the $2^d - 1$ imaginary hypercubes that cover it. Let m and m' be the sizes of the set of hypercubes now sharing a facet with f and f' respectively, hence $m = |\gamma(f)|$ and $m' = |\gamma(f')|$. As each adjacency can be charged to a characteristic point of a hypercube from one of the two sets and no such point gets charged more than a constant number of times (see the proof of Lemma 4.3), the number of adjacencies is $O(m + m')$. Because every hypercube from $\gamma(f)$ is adjacent to at least one hypercube from $\gamma(f')$, we can even conclude that the number of adjacencies is $\Theta(m + m')$.

Using the observation that one of two adjacent hypercubes has a characteristic point in

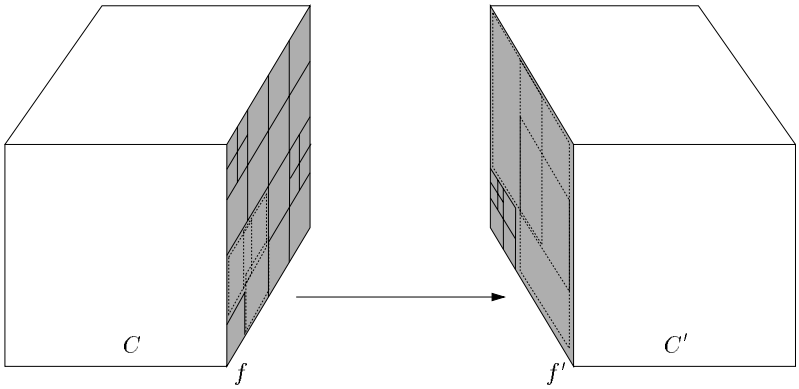

Figure 8: The grey facets f of C and f' of C' are glued onto each other. Pairs of regions from C and C', or more specifically, pairs of regions sharing a facet with f and f', become adjacent.

the interior of a facet of the other, we find the following simple strategy for computing all adjacencies: determine for each characteristic point of a hypercube in $\gamma(f)$ the hypercube in $\gamma(f')$ containing it, and vice versa. Note that we can restrict ourselves to querying with characteristic points on the joined facets f and f'. Keeping in mind that our ultimate aim is to compute the pairwise adjacencies of regions from V_{W} we can just as well replace this strategy by: determine for each characteristic point of a hypercube in $\gamma(f)$ the region (in C') containing it, and vice versa. The point location structure of the decomposition allows to find the region containing a query point in $O(\log n)$ time. We observe that all our query points lie on the common boundary of at least two regions. To overcome this complication, we (symbolically) move the query points from $\gamma(f)$ perpendicularly away from f (and C), thereby moving a query point in the interior of a facet of an adjacent region into the interior of that region. The queries with the $O(m)$ characteristic points of the hypercubes in $\gamma(f)$ in $O(m \log n)$ time. Likewise, we can solve the reverse problem in $O(m' \log n)$ time. No adjacency will be reported more than a constant number of times.

The preceding analysis shows that the time required to compute the k adjacencies created by uniting C and C' is bounded by $O(k \log n)$. Moreover, the computation of all K adjacencies resulting from joining the 2^d equally-sized hypercubes into one larger hypercube takes $O(K \log n)$ time. The sets $\gamma(f)$ for each of the facets f of the newly obtained large hypersphere are trivially computable from the appropriate sets stored with the facets of the 2^d sub-hypercubes. Combining the time bounds for both types of joining steps and the $O(n)$ bound on the size of E_{W} given by Lemma 4.3 leads to the following result.

Lemma 4.4 *The set E_{W} of pairwise adjacencies of regions in V_{W} is computable in $O(n \log n)$ time.*

We have found linear-size cc-partitions for workspaces of arbitrary dimension. These partitions are computable in $O(n \log n)$ time. Substitution of these computations into the first step of algorithm LODMOT yields the following result by Theorem 3.9.

Theorem 4.5 *The low density motion planning problem can be solved in $O(n \log n)$ time.*

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 semi-algebraic 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 minimal enclosing hypersphere radius 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 $O(n)$ [30], 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 partition 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.

We have shown that optimal $O(n)$ size workspace partitions exist in any dimension d. The partitions are computable in $O(n \log n)$ time. These results lead to $O(n \log n)$ algorithms for solving the low density motion planning problem. The bounds show that our approach of decomposing some lower-dimensional subspace of the configuration space (subject to some constraints) results in efficient solutions to the low density motion planning problem.

It is 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 $C = \mathrm{W} \times D$. We have already mentioned (see Subsection 3.2) the motion planning problem for a robot moving in a three-dimensional world while its motion is confined to a plane, e.g. a factory floor. The results from this paper are almost immediately applicable to this problem [33]. Other possible extensions include motion planning with moving obstacles, multiple robots, and anchored robot arms.

The dynamic version of the low density motion planning problem has been studied by Berretty et al. [6]. They consider a setting in which the obstacles in the workspace move at constant speed along polyline paths. The workspace is assumed to satisfy the low density property at any time. The authors use the ideas from Subsection 3.1 to find a nearly-optimal $O(n^2 \alpha(n) \log^3 n)$ solution to the problem.

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 a cylindrical decomposition of the free space seem applicable if the workspace W is a projective subspace of each of the spaces C_i; in that case, W^c is a valid base space. Vleugels [34] has shown how to restrict the search for a free path for two robots in their composite configuration space C to a collection

of lower-dimensional subspaces of C, such that a path exists in the subspaces whenever one exists in C. The approach leads to $O(n \log n)$ and $O(n \log^2 n)$ algorithms for two robots in two- and higher-dimensional workspaces respectively, even though the complexity of the free spaces can be $\Omega(n^2)$.

For most industrial robot arms, the links close to the hand—the minor axes—are considerably shorter than the links close to the base—the major axes. Consider an f-link robot arm of which the m minor axes 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 at the moment how to compute base partitions in C'.

References

[1] P.K. Agarwal, M.J. Katz, and M. Sharir, Computing depth orders for fat objects and related problems, *Computational Geometry: Theory and Applications* **5** (1995), pp. 187-206.

[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] 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.

[4] M. de Berg, Linear size binary space partitions for fat objects, Technical Report, Dept. of Computer Science, Utrecht University, to appear.

[5] M. de Berg, M. Katz, A.F. van der Stappen, and J. Vleugels, Realistic input models for geometric algorithms, *Proc. 13th Ann. ACM Symp. on Computational Geometry* (1997), pp. 294-303.

[6] R.-P. Berretty, M. Overmars, and A.F. van der Stappen, Dynamic motion planning in low obstacle density environments, *Proc. Workshop on Algorithms and Data Structures (WADS'97)* (1997).

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

[8] B. Chazelle and L. Guibas, Fractional cascading I: A data structuring technique, *Algorithmica* **1** (1986), pp. 133-162.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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).

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

[20] M.H. Overmars and A.F. van der Stappen, Range searching and point location among fat objects, *Journal of Algorithms* **21** (1996), pp. 629-656.

[21] 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).

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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).

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

[34] J. Vleugels, *On fatness and fitness—realistic input models for geometric algorithms*, Ph.D. Thesis, Dept. of Computer Science, Utrecht University (1997).

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

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