Proc. IEEE International Conference on Robotics and Automation, Atlanta, 1993, Vol. 1, pp.297–304.

Efficient Algorithms for Exact Motion Planning amidst Fat Obstacles*

A. Frank van der Stappen† Dan Halperin‡ Mark H. Overmars†

†Dept. of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
‡Robotics Laboratory, Dept. of Computer Science, Stanford University, Stanford, CA 94305, USA

Abstract

The complexity of exact motion planning algorithms highly depends on the complexity of the robot's free space, i.e., the set of all collision-free placements of the robot. Theoretically, the complexity of the free space can be very high. In practice, the complexity of the free space tends to be much smaller. We show that, under some realistic assumptions, the complexity of the free space of a robot moving amidst fat obstacles is linear in the number of obstacles. The complexity results lead to efficient algorithms for motion planning amidst fat obstacles: we show that the $\mathcal{O}(n^5)$ motion planning algorithm by Schwartz and Sharir runs in $\mathcal{O}(n^2)$ time if the obstacles are fat. Finally, we modify the algorithm to improve the running time to $\mathcal{O}(n \log^2 n)$.

1 Introduction

Autonomous robots are one of the ultimate goals in the field of robotics. An autonomous robot must accept high-level descriptions of tasks and execute these tasks without further intervention from its environment.

An obvious task for an autonomous robot would be to move from one place to another. While moving, the robot must avoid collision with the obstacles that are present. The problem of finding such a collision-free motion is referred to as the *motion planning problem* (see [3] for an overview of the field's state-of-the-art). We consider the following version of the general motion planning problem.

> Given a robot $\mathcal{B}$ moving amidst a collection of obstacles $\mathcal{E}$, and an initial placement Z_0 and a final placement Z_1 for $\mathcal{B}$, find a continuous motion for $\mathcal{B}$ from Z_0 to Z_1 during which the robot avoids collision with the obstacles, or report that no such motion exists.

A robot $\mathcal{B}$ moves around in a *workspace* W, usually being the Euclidean space of dimension two or three (R^2 or R^3). The collection of obstacles $\mathcal{E}$ is a closed, possibly unbounded, subset of W. A connected component E of the set $\mathcal{E}$ is referred to as an *obstacle*.

One way of formalizing the motion planning problem is to transform the problem into the problem of planning the motion of a point robot in the robot's *configuration space*. The configuration space C of the robot is the set of parametric representations of robot placements. The parameters that are needed to specify a robot placement are referred to as the *degrees of freedom* of the robot. A point $Z \in C$ is a placement. The set of points in W covered by $\mathcal{B}$ placed at Z is denoted by $\mathcal{B}[Z]$.

A point Z in the configuration space C that corresponds to a placement of the robot $\mathcal{B}$ in which it intersects no obstacle is called a *free placement*. The *free space* FP is the set of all free placements of $\mathcal{B}$, hence

$$\mathrm{FP} = \{\, Z \in C \mid \mathcal{B}[Z] \cap \mathcal{E} = \emptyset \,\}.$$

A *collision-free path* for a robot $\mathcal{B}$ from Z_0 to Z_1 is a continuous map $\tau : [0,1] \rightarrow \mathrm{FP}$, with $\tau(0) = Z_0$ and $\tau(1) = Z_1$. So, the problem of motion planning is equal to the problem of finding a continuous curve completely lying inside the free portion FP of the configuration space. The effort that is required to find such a curve is obviously highly dependent on the complexity of the free space FP.

The complexity of FP, as we will see in the next section, is determined by the number of multiple contacts of the robot $\mathcal{B}$ with the obstacles. A multiple contact of $\mathcal{B}$ is a placement in which it touches more than one obstacle feature (e.g. edge, corner). Unfortunately, the number of multiple contacts can be very high. If n is the number of obstacle features and f is the number of degrees of freedom of the robot (i.e., the dimension of C) and the number of robot features is bounded by some constant, then this complexity can be $\Omega(n^f)$. This leads to high worst case time bounds for motion planning algorithms. Fortunately, in many practical situations, the number of multiple contacts is much smaller and, hence, these algorithms might become feasible [2] (assuming that they are sensitive to the size of FP). A study of properties that limit the number of multiple contacts for the robot (and hence the complexity of FP) is therefore of obvious importance.

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

2 Multiple contacts

The complexity of a collection of connected sets is determined by the complexity of its boundaries. Since FP is a collection of connected subsets of the configuration space C, the complexity of FP is determined by the complexity of the boundary of FP. The set FP is an open subset of C, so its boundary BFP is obtained by subtracting FP from its closure $\overline{\mathrm{FP}}$, yielding:

$$\mathrm{BFP} = \{\, Z \in C \mid \mathcal{B}[Z] \cap int(\mathcal{E}) = \emptyset \ \wedge\ \mathcal{B}[Z] \cap \mathcal{E} \neq \emptyset \,\},$$

where $int(\mathcal{E})$ is the obstacle set without its boundaries. So, the boundary BFP of FP is the set of placements in which the robot only touches the obstacles and not intersects their interiors. We examine the set of such contact placements, since it determines the complexity of FP.

Let us assume that the configuration space C has dimension f. The set of all robot placements in which a specific robot feature is in contact with a specific obstacle feature is an (f-1)-dimensional subset, or hypersurface, in C. Each combination of a robot feature and an obstacle feature defines such a hypersurface. If the total number of features (the complexity) of the robot and the obstacles is $\mathcal{O}(1)$ and $\mathcal{O}(n)$ respectively, then the total number of hypersurfaces equals $\mathcal{O}(n)$.

The contact hypersurfaces can obviously intersect each other. A point on the intersection of j hypersurfaces corresponds to a j-fold contact of the robot $\mathcal{B}$. The intersection of j hypersurfaces is an (f-j)-dimensional subspace of the configuration space. All points in this subspace correspond to placements in which the same j contacts exist. Any j-tuple of hypersurfaces can theoretically be involved in such an intersection, so the total number of intersections is $\mathcal{O}(n^j)$. An f-fold contact clearly defines a point in C; contacts that involve more than f feature pairs only appear occasionally and can be regarded as an f-fold contact placement in which the robot accidentally touches one or more additional obstacle features.

The complexity of BFP, and, hence, of FP itself, is determined by the total number of subspaces defined by the $\mathcal{O}(n)$ contact hypersurfaces that constitute the boundary. This number of subspaces equals the sum of the number of hypersurfaces and the number of multiple contacts for the robot $\mathcal{B}$, provided that each intersection of hypersurfaces consists of only a constant number of connected sets. The latter requirement will generally be satisfied. Theorem 2.1 states this result on the relation between the free space complexity and the number of multiple contacts.

Theorem 2.1 *Let $\mathcal{B}$ be a constant complexity robot with f degrees of freedom and let $\mathcal{E}$ be a set of obstacles with total complexity $\mathcal{O}(n)$. Then the complexity of the free space of the robot $\mathcal{B}$ is $\mathcal{O}(n + \sum_{2 \leq j \leq f} \mathcal{N}(j))$, where $\mathcal{N}(j)$ is the number of j-fold contacts for the robot $\mathcal{B}$.*

3 Fat obstacles

In many practical cases the relative positions and the shapes of the obstacles are such that the number of multiple contacts for the robot $\mathcal{B}$ is very low. Obstacles that are relatively far apart compared to the size of $\mathcal{B}$, clearly result in less multiple contacts for $\mathcal{B}$ than obstacles that are cluttered. Similarly, obstacles that have no long and skinny parts -we will call such obstacles *fat*- induce less multiple contacts than obstacles that do have such parts.

Fatness turns out to be a very interesting property in computational geometry. It may lead to improved combinatorial bounds [1, 4] and to efficient algorithms [6, 11]. Fatness is a natural concept in motion planning, since in many practical situations, the obstacles will be fat to a certain extent. We will see in this paper that fatness of the obstacles leads to an interesting combinatorial bound *and* to an efficient motion planning algorithm.

Our definition of fatness [10] in a d-dimensional Euclidean workspace involves a set U_E of spherical regions. The choice for spherical regions is rather arbitrary; a choice for a different "compact" (e.g. cubic) region would lead to similar results.

Definition 3.1 [U_E]
Let $E \subseteq R^d$ be an obstacle. The set U_E is defined as the set of all spherical regions with center inside E that do not fully contain E.

We define fatness in such a way that obstacles are not only "compact" but also do not have extremely thin protuberances. The definition of fatness involves some positive number k, which is a measure for the actual fatness of the obstacle. If k is increased then the obstacle is allowed to be less fat. For obstacles with a boundary with infinitesimally thin protuberances (e.g. line segments) it is impossible to find such a k, so these obstacles can never be fat.

Definition 3.2 [k-fatness]
Let $E \subseteq R^d$ be an obstacle and let k be a positive constant. The obstacle E is k-fat if for all $S \in U_E$:

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

Informally, an obstacle E is k-fat if the part of any spherical region S (with a boundary that intersects E and its center inside E) covered by E is at least a $\frac{1}{k}^{\mathrm{th}}$ of S.

Let us consider some examples of fat shapes. Spherical obstacles are obviously fat; a sphere in d-space has maximal achievable 2^d-fatness. Other shapes that are fat (in R^2) include squares, rectangles with bounded length-width ratio, triangles with angle restriction, and many more irregularly shaped objects as well. In R^3, we can think of cubes, equilateral tetrahedra, boxes with bounded ratios of length, width, and height, etc.

4 The complexity of FP

Let us now return to determining the complexity of the free space for motion planning amidst fat obstacles. We will show that, under some realistic assumptions, the number of multiple contacts for the robot is linear in the number of obstacle features. Theorem 2.1 then leads to linear complexity of the free space.

We consider the situation where a robot $\mathcal{B}$ moves amidst k-fat obstacles $E \subseteq \mathcal{E}$. The robot $\mathcal{B}$ is assumed to be not too big compared to the obstacles. Let the diameter of the smallest minimal enclosing hypersphere of any obstacle be d_{min}. The diameter $d_{\mathcal{B}}$ of the robot is constrained by $d_{\mathcal{B}} \leq b \cdot d_{min}$, where b is some positive constant. This assumption regarding the size of the robot is not very restrictive: it basically rules out the situation where the robot $\mathcal{B}$ is so large that it would make the obstacles into point obstacles relative to its own size.

We assume that the number of features of the robot $\mathcal{B}$ is bounded by a constant and the number of features of the obstacle set $\mathcal{E}$ is n. As a consequence, the total number of hypersurfaces is $\mathcal{O}(n)$. The hypersurfaces are assumed to be algebraic surfaces of bounded degree, so that the number of intersections of two hypersurfaces is also bounded by a constant. This requirement for the degree of the hypersurfaces mainly means that the boundary of the robot and the obstacles must not be too irregularly shaped. As the hypersurfaces are of bounded degree this implies that the total complexity of the free space FP is bounded by $\mathcal{O}(n^f)$. Each obstacle E in $\mathcal{E}$ is assumed to have a constant number of features.

All results in the remainder of this paper will hold under the assumptions mentioned above. In the sequel, the obstacle with the smallest enclosing spherical region will be denoted by E_{min}.

As a first step in finding an upper bound on the number of multiple contacts for $\mathcal{B}$, we consider the proximity of a k-fat obstacle. It is obvious that two obstacles that are far (more than $d_{\mathcal{B}}$) apart can not be involved in any multiple contact of $\mathcal{B}$. Hence, obstacles that do cause such a contact must lie in each other's proximity.

A strategy for proving a linear upper bound on the number of multiple contacts for the robot $\mathcal{B}$ could be to prove that the number of multiple contacts involving a certain obstacle is only constant. Straightforward application of this strategy, however, would yield no result. If we have a situation with $\mathcal{O}(n)$ equally sized k-fat obstacles and one much larger k-fat obstacle, then all $\mathcal{O}(n)$ smaller obstacles might lie in the proximity of the large obstacle. Note, however, that this strategy contains some redundancy: a multiple contact is counted more than once.

To avoid counting a single multiple contact more than once, we only count the number of larger obstacles that can participate in a multiple contact involving E. By starting with the smallest obstacle and repeatedly considering the next smallest obstacle we count each multiple contact of the robot exactly once. It turns out that the number of such multiple contacts involving any obstacle E is constant.

Any k-fat obstacle E' that participates in some multiple contact with a given k-fat obstacle E must lie close to E. Lemma 4.1 (see [10] for all proofs of results in this section) states that the number of obstacles that lie in the proximity of the obstacle E_{min} is bounded by a constant.

Lemma 4.1 *Let $\mathcal{E} \subseteq \mathrm{W} = R^d$. The number of obstacles $E \subseteq \mathcal{E}$ that lie close enough to E_{min} so that $\mathcal{B}$ can touch E_{min} and E simultaneously is at most $2^d \cdot k \cdot (b+1)^d$.*

It is not surprising that the constant in Lemma 4.1 increases when the obstacles become less fat (k increases) or when the diameter of $\mathcal{B}$ increases (b increases).

The proximity result given in Lemma 4.1 is the key to successful application of the proof strategy outlined above. The strategy results in an overall linear number of multiple contacts, as stated in Theorem 4.2.

Theorem 4.2 *Let $\mathcal{E} \subseteq \mathrm{W} = R^d$. For each j $(2 \leq j \leq f)$, the number of j-fold contacts $\mathcal{N}(j)$ of the robot $\mathcal{B}$ is linear in the number of obstacles: $\mathcal{O}(n)$.*

The $(f\text{-}j)$-dimensional subspace defined by a single j-fold contact is not necessarily connected. Figure 1 shows an example for f=3 and j=2, where it is impossible for the robot to move from Z_0 to Z_1 without losing contact with either the upper or the lower obstacle feature. The 1-dimensional subspace induced by the contact with both features is therefore non-connected. Our assumption that

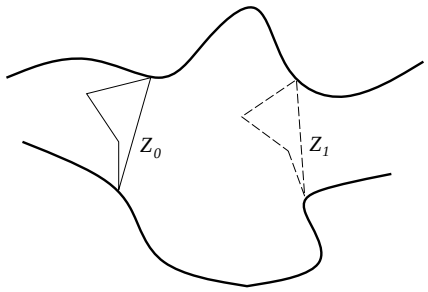

Figure 1: **No double contact motion from Z_0 to Z_1**

all contact hypersurfaces are of bounded degree, however, implies that the number of connected subspaces induced by a single multiple contact is bounded by some constant. Hence, the requirement for applicability of Theorem 2.1 is satisfied, resulting in Corollary 4.3.

Corollary 4.3 *Let $\mathcal{E} \subseteq \mathrm{W} = R^d$. The free space for the robot $\mathcal{B}$ moving amidst $\mathcal{E}$ has linear complexity.*

Sifrony and Sharir [9] present an algorithm for planning the motion of a ladder amidst polygonal obstacles running in $\mathcal{O}(K \log n)$, where K is the number of obstacle feature pairs that are less than the length of the ladder apart. Their algorithm is easily seen to benefit from our linear complexity result. A direct, and easy to prove consequence of this result is that $K = \mathcal{O}(n)$ in a workspace with k-fat obstacles, instead of $K = \mathcal{O}(n^2)$ in the case of arbitrary obstacles. For most other motion planning algorithms the possible benefit is less obvious, since their complexities are given in terms of n (the number of obstacle features) only. A more extensive study of these algorithms is required to conclude that they benefit from the linear complexity result.

5 Efficient motion planning

Motion planning algorithms preprocess the free space FP for answering path finding queries. The linear complexity result for FP does not directly imply that the outcome of the preprocessing, a representation of FP, has linear complexity as well. Moreover, if an algorithm succeeds in supplying a linear complexity representation, then it may still take far more time to compute this representation.

The *cell decomposition* approach partitions FP into a finite number of simple connected cells, such that planning a motion between any two placements within a single cell is straightforward and such that uniform crossing rules can be defined for $\mathcal{B}$ crossing from one cell into another. Each cell defines a node in the *connectivity graph* CG. Two nodes in CG are connected by an edge if their corresponding cells share a common boundary allowing direct crossing of the robot $\mathcal{B}$. Given the connectivity graph CG, the problem of motion planning is reduced to a graph problem: find a sequence of cells (nodes of CG) connecting the cells containing the placements Z_0 and Z_1. The properties of the cells guarantee that it is easy to transform the sequence into an actual path for $\mathcal{B}$.

Schwartz and Sharir [7] apply the cell decomposition technique to obtain an $\mathcal{O}(n^5)$ algorithm for planning the motion of a ladder $\mathcal{B}$ moving amidst polygonal obstacles $\mathcal{E}$ in the plane. Their method decomposes $\mathrm{W} = R^2$ into so-called *noncritical* regions, then lifts these regions into three-dimensional *cells* (in $C = R^2 \times [0, 2\pi)$), and finally captures the adjacency of these cells in a connectivity graph. We will go into more detail on each of these steps and, while doing so, focus on the consequences of fatness for each of these steps. We emphasize that we will not give an extensive explanation of the ladder algorithm. A more extensive explanation is given in [11].

The noncritical regions in the robot's workspace $\mathrm{W} = R^2$ are defined by *critical curves*. The meaning of these curves is not important in our analysis, so we restrict ourselves to summarizing the different types of critical curves. We adopt the classification of the critical curves used in [3]. Let P and Q be the endpoints of the ladder $\mathcal{B}$ and let $d_{\mathcal{B}}$ be its length. Choose P as the robot's reference point. Figure 2 illustrates the various critical curve types.

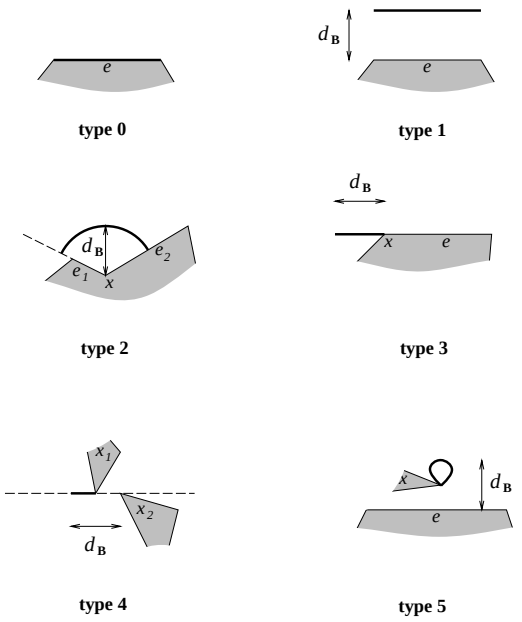

Figure 2: **The six types of critical curves.**

- An obstacle edge is a critical curve of **type 0**.
- Let e be an obstacle edge. The line segment at a distance $d_{\mathcal{B}}$ from e is a critical curve of **type 1**. The length of the critical curve equals the length of e.
- Let x be an obstacle corner and let e_1 and e_2 be the edges emerging from x. The circular arc with radius $d_{\mathcal{B}}$, centered at x, and running between the half-lines starting at x and containing the edges e_1 and e_2 respectively, is a critical curve of **type 2**.
- Let x be a convex obstacle corner and let e be one of the edges emerging from x. The line segment traced out by P while $\mathcal{B}$ slides along e, so that Q touches e and x touches $\mathcal{B}$, is a critical curve of **type 3**.
- Let x_1 and x_2 be convex obstacle corners such that the line passing through x_1 and x_2 is tangent to the obstacle set $\mathcal{E}$ in both x_1 and x_2. The line segment traced out by endpoint P, while $\mathcal{B}$ slides along x_1 and x_2, is a critical curve of **type 4**. Note that the distance from x_1 to x_2 must be less than $d_{\mathcal{B}}$.
- Let x be a convex obstacle corner and let e be an obstacle edge such that x is not an endpoint of e. The curve traced out by P while Q slides along e and while $\mathcal{B}$ remains in contact with x, is a (fourth degree) critical curve of **type 5**. Note again that the distance from x to e must be less than $d_{\mathcal{B}}$.

The (intersecting) critical curves partition $\mathrm{W} = R^2$. The part of a critical curve between two points of intersection with other critical curves is called a *critical curve section*. A position (x, y) of the robot $\mathcal{B}$ is *admissible* if there exists an orientation θ, such that $(x, y, \theta) \in \mathrm{FP}$. A noncritical region is a maximal subset of admissible robot positions intersecting no critical curves. Hence, the critical curves determine a set of noncritical regions in W.

Let $\Theta_{x,y} = \{\, \theta \,|\, (x, y, \theta) \in \mathrm{FP} \,\}$ be the set of free orientations of $\mathcal{B}$ with P fixed at a point (x, y) in a noncritical region R. The set $\Theta_{x,y}$ consists of a finite number of open maximum connected intervals. For each such interval $(\theta_1, \theta_2) \in \Theta_{x,y}$, both the robot placement (x, y, θ_1) and the robot placement (x, y, θ_2) are placements in which the robot touches the obstacle set. The unique stop touched by $\mathcal{B}$ in the contact placement (x, y, θ_1) (resp. (x, y, θ_2)) is denoted by $s(x, y, \theta_1)$ (resp. $s(x, y, \theta_2)$). The set of all pairs $[s(x, y, \theta_1), s(x, y, \theta_2)]$ such that $(\theta_1, \theta_2) \in \Theta_{x,y}$ is referred to as $\sigma(x, y)$. The critical curves are defined so that for each pair of points (x, y) and (x', y') in a single noncritical region R, the sets $\sigma(x, y)$ and $\sigma(x', y')$ are equal [7]. In the sequel, we use the abbreviation $\sigma(R) = \sigma(x, y)$, where (x, y) is any point in R. Each pair of contact positions $[s_1, s_2] \in \sigma(R)$ defines a cell in the cell decomposition of FP.

Schwartz and Sharir's method first computes all critical curves, and then all intersections of the curves, resulting in a collection of intersection points and a collection of critical curve sections. With each intersection point, we store the critical curve sections that are incident at this intersection point. Each critical curve section β separates two regions; we arbitrarily call one of the regions $\mathit{left}(\beta)$ and the other one $\mathit{right}(\beta)$. Next, we compute $\sigma(\mathit{left}(\beta))$ and $\sigma(\mathit{right}(\beta))$, define a connectivity graph node for each cell $[s_1, s_2]$ induced by the regions $\mathit{left}(\beta)$ and $\mathit{right}(\beta)$, and build the adjacency relation (based on the adjacency of the corresponding cells) between the nodes induced by both regions. (Note that each node is generated by a single critical curve section.) If we repeat this procedure for every critical curve section, each cell is represented in the connectivity graph as many times as there are critical curve sections bordering the region that induced the cell. All nodes that correspond to the same cell are circularly connected.

5.1 Complexity of the cell decomposition

We recall that the number of obstacle edges and corners is $\mathcal{O}(n)$. First, we observe that the number of type 0-3 curves is not influenced by the fatness of the obstacles and is $\mathcal{O}(n)$ in both the arbitrary and the fat case.

In the general case of arbitrary polygonal obstacles, the number of type 4 curves is $\mathcal{O}(n^2)$ since each pair of corners may define such a curve. The same number applies to type 5 curves since each pair of one edge and one (convex) corner may induce such a curve. As a result, the total number of critical curves is $\mathcal{O}(n^2)$. Each pair of curves may intersect, so that there can be up to $\mathcal{O}(n^4)$ intersections, and, hence, $\mathcal{O}(n^4)$ critical curve sections.

Things change if we assume the obstacles to be k-fat. Let us first observe an important property of all critical curves, following from the definitions of the curves.

Property 5.1 *Each point on a critical curve is less than the length $d_{\mathcal{B}}$ of the ladder away from the obstacle features (corners, edges) that define this curve.*

Another important tool in the analysis of the "fat" case is the following generalization of Lemma 4.1.

Lemma 5.2 *Let $\mathcal{E} \subseteq \mathrm{W} = R^2$ be a set of polygonal obstacles and let $c, c' \geq 0$ be two constants. Then the number of obstacles $E \subseteq \mathcal{E}$ having non-empty intersection with any square region C with side length $c \cdot d_{min} + c' \cdot d_{\mathcal{B}}$ is bounded by a constant.*

We count the number of type 4 and 5 curves with at least one of their defining features on the smallest obstacle E_{min}. Let f_1 be a feature of E_{min}. If this feature induces a type 4 or type 5 curve together with some other feature f_2, then f_2 must be less than a distance $d_{\mathcal{B}}$ away from f_1, since $\mathcal{B}$ must be in simultaneous contact with both features. Moreover, such a feature f_2 will have non-empty intersection with a square region C with side length $d_{min} + 2d_{\mathcal{B}}$, that is obtained by first taking a square region with side length d_{min} enclosing E_{min} and subsequently pushing its sides outward by a distance $d_{\mathcal{B}}$ (see Figure 3). Lemma

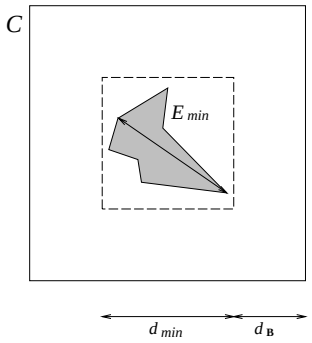

Figure 3: **The square region C.**

5.2 with c=1 and c'=2 shows that the number of obstacles having non-empty intersection with C is bounded by a constant. Since all obstacles have constant complexity, the number of obstacle features in C is constant, and, hence, the number of candidates for f_2 is bounded by a constant. The number of type 4 or type 5 curves induced by a feature f_1 is therefore bounded by a constant. The constant complexity of E_{min} then guarantees that the number of type

4 or type 5 curves involving E_{min} is constant. Repeating these arguments for every next smallest obstacle, results in a total of $\mathcal{O}(n)$ type 4 and type 5 curves. Combination with the linear number of type 0-3 curves yields Lemma 5.3.

Lemma 5.3 *The number of critical curves in the workspace is $\mathcal{O}(n)$.*

Let β be a critical curve that is induced by at least one of the features f_1 of E_{min}. The distance from f_1 to a possible intersection point p of β with some other critical curve β' is at most $d_{\mathcal{B}}$ by Property 5.1. Let F be the set of features defining β'. Again by Property 5.1, the distance from p to all $f_2 \in F$ is bounded by $d_{\mathcal{B}}$. The distance from feature f_1 to all $f_2 \in F$ is at most $2d_{\mathcal{B}}$. Moreover, these features f_2 will have non-empty intersection with a square region C with side length $d_{min} + 4d_{\mathcal{B}}$, that is obtained by first taking a square region with side length d_{min} enclosing E_{min} and subsequently pushing the sides outward by a distance $2d_{\mathcal{B}}$. Lemma 5.2 (c=1,c'=4) and the constant complexity of the obstacles establish that the number of candidate members of F is bounded by a constant, and, hence, the number of critical curves β' intersecting β. Since all critical curves have low degree, the number of intersection points on β is bounded by a constant. Repeating these arguments for each of the critical curves induced by E_{min} and then for each next smallest obstacle, yields Lemma 5.4.

Lemma 5.4 *The number of critical curve sections in the workspace is $\mathcal{O}(n)$.*

We have explained that the number of connectivity graph nodes added by the critical curve section β equals the number of cells induced by the region $left(\beta)$ plus the number of cells induced by the region $right(\beta)$. If β is a type 0 curve, only one of the two regions is noncritical, in all other cases both regions will be noncritical. A region that is *not* noncritical will induce no graph nodes.

We analyze the number of cells induced by a single noncritical region R. In Schwartz and Sharir's method, each pair $[s_1, s_2] \in \sigma(R)$ defines a cell. Hence, the number of cells induced by a noncritical region R is determined by the number of pairs in $\sigma(R)$, each pair in $\sigma(R)$ consisting of two different contact placements for $\mathcal{B}$ with P fixed at some point in R. The number of cells induced by R is therefore determined by the number of different contact placements for $\mathcal{B}$ when we fix its endpoint P at some point (x, y) in R and vary its orientation θ. In the case of arbitrary polygonal obstacles, we can easily construct examples where the robot can touch any of the $\mathcal{O}(n)$ obstacle features, each at a different orientation θ. A noncritical region R can therefore induce $\mathcal{O}(n)$ cells. Since the number of noncritical regions is $\mathcal{O}(n^4)$, we obtain a total number of $\mathcal{O}(n^5)$ cells. As before, things are different in our fat setting. It is easy to see that an obstacle feature f touched by $\mathcal{B}$ with P fixed at (x, y) in some contact position has non-empty intersection with the square region C with side length $2d_{\mathcal{B}}$ and center (x, y) (intersection point of the diagonals). The region C satisfies again the constraints of Lemma 5.2 (c=0,c'=2). Hence, the number of obstacle features f that can be touched by $\mathcal{B}$ with P fixed at (x, y) is bounded by a constant. Moreover, the number of pairs in $\sigma(R)$ is bounded by a constant. This leads to the result stated in Lemma 5.5.

Lemma 5.5 *Each noncritical region in the workspace induces only $\mathcal{O}(1)$ cells in the configuration space.*

Lemma 5.5 shows that each critical curve β adds at most twice a constant number of nodes to the connectivity graph. By Lemma 5.4, we conclude that the total number of nodes is $\mathcal{O}(n)$. Let N be a node added by a critical curve β and assume without loss of generality that N corresponds to a cell induced by $left(\beta)$. Then N can be adjacent to the nodes added by β and corresponding to cells induced by $right(\beta)$, and to nodes that N is circularly connected to and correspond to the same cell. By Lemma 5.5 the first set contains only a constant number of nodes, whereas the second set contains at most two nodes. Hence, each cell is adjacent to $\mathcal{O}(1)$ nodes, resulting in a total of $\mathcal{O}(n)$ graph edges.

Theorem 5.6 *The connectivity graph corresponding to the cell decomposition of the free space of a ladder moving amidst k-fat obstacles has $\mathcal{O}(n)$ nodes and edges.*

5.2 Computing the cell decomposition

Although the complexity of the connectivity graph of the cell decomposition is $\mathcal{O}(n)$, a straightforward application of Schwartz and Sharir's method would result in $\mathcal{O}(n^2)$ time to compute the decomposition. This bound is due to three steps in the algorithm: a first step where all critical curves are computed, a second step where all critical curve sections are computed, and a third step where all cells induced by a single noncritical region are determined.

We have shown in the previous subsection that the number of type 4 or 5 curves is linear in the case of fat obstacles. Each of these curves is determined by two features. We could naively try all possible pairs of features to find out which pairs generate a curve. This strategy would require $\Omega(n^2)$ time to find the $\mathcal{O}(n)$ curves. Instead we should use the knowledge that only two features that lie close enough to each other can define a curve. Our approach is to start with the smallest obstacle E_{min} and compute all critical curves involving a feature f_1 of E_{min}. The analysis in the previous subsection shows that obstacles supplying the second feature f_2 for a type 4 or 5 curve) have non-empty intersection with a square region C with

side length $d_{min} + 2d_{\mathcal{B}}$, obtained by growing an enclosing square of E_{min} by the robot's diameter $d_{\mathcal{B}}$. If we first compute this set of obstacles, then we are left with a constant number of candidates for f_2. From this constant size set, we can compute the critical curves involving E_{min} in constant time. The time required for computing these curves is therefore determined by the time to find the set of candidates. We subsequently discard E_{min} and repeat the procedure with the remaining obstacles.

The problem that we are now left with is the problem of finding for each obstacle E all larger obstacles that have non-empty intersection with a square region C enclosing E. The region C is not uniqely defined; it can be chosen to have any orientation. Finding a solution to the problem is simplified if we choose all square regions to have the same (axis-parallel) direction. The resulting axis-parallel square intersection problem can be solved using the following theorem from [5].

Theorem 5.7 *Given a set of n non-intersecting line segments in the plane, we can store them using $\mathcal{O}(n \log n)$ storage such that those K segments visible in a given axis-parallel square can be determined in $\mathcal{O}(K + \log^2 n)$ time. The structure is dynamic and can be updated in $\mathcal{O}(\log^2 n)$ time.*

Let V_E be the set of obstacles that are larger than E and have non-empty intersection with the square region enclosing E. Our approach is the following. We order the, say n, obstacles in $\mathcal{E}$ by increasing size of their enclosing squares: $E_1 ... E_n$, such that E_1 is the obstacle with the smallest enclosing square. Assume that we are given the data structure of the theorem storing the boundary edges of the obstacles $E_{i+1} ... E_n$. We will now determine the set V_{E_i}, by first computing the square region C_i related to E_i (this is an axis-parallel enclosing square of E_i grown by $d_{\mathcal{B}}$). Then we can search with the square region C_i in the data structure containing the boundary edges of the obstacles $E_{i+1} ... E_n$ and find the constant number of intersecting obstacles ($K = \mathcal{O}(1)$ by Lemma 5.2) in $\mathcal{O}(\log^2 n)$ time. Subsequently, we add the constant number of boundary edges of obstacle E_i to the data structure in $\mathcal{O}(\log^2 n)$ time. Next, the entire procedure is repeated for obstacle E_{i-1}. Using this approach, we can determine all sets V_E in time $\mathcal{O}(n \log^2 n)$.

Having the sets V_E at our disposal, we can efficiently compute all critical curves. We start with the smallest obstacle E_{min} and compute all critical curves involving a feature f_1 of E_{min}. If such a critical curve involves a second feature f_2, then f_2 will be feature of one the members of $V_{E_{min}}$. The constant size of all the sets V_E certifies that, if we repeat this procedure for every next smallest obstacle, the total number of feature pairs considered is $\mathcal{O}(n)$, and therefore of the same order of magnitude as the number of critical curves (Lemma 5.3).

After having computed the critical curves in $\mathcal{O}(n \log^2 n)$ time, we encounter a similar problem when we compute the curve intersections (and the resulting critical curve sections). We could naively take each critical curve and intersect it with all other critical curves, but this would require $\mathcal{O}(n^2)$ computations to find only $\mathcal{O}(n)$ intersections. Instead we should use the knowledge that the features that define two intersecting curves must lie close enough to each other, i.e., less than $2d_{\mathcal{B}}$ apart. Again we start with the smallest obstacle E_{min} and compute all critical curves β' intersecting a critical curve β induced by at least one feature f of E_{min}. The set of features F defining β' must have non-empty intersection with the region C obtained by taking an enclosing square of E_{min} and growing it by $2d_{\mathcal{B}}$. Applying the same ideas as above for these (larger) boxes results in a sequence of constant size sets V'_E that can be obtained in $\mathcal{O}(n \log^2 n)$ time. With these sets we can find all critical curve intersections, and, hence, all critical curve sections, in linear time.

For each of the $\mathcal{O}(n)$ critical curve sections β, we have to compute $\sigma(left(\beta))$ and $\sigma(right(\beta))$. Computing $\sigma(left(\beta))$ is done by taking a point $(x, y) \in left(\beta)$, fixing the robot's endpoint P at (x, y), and reporting all features that can be touched by $\mathcal{B}$ and the orientations in which they are touched. We could naively try all n features, but then it would require $\mathcal{O}(n^2)$ to find all connectivity graph nodes. Instead we should use the knowledge that a feature that can be touched by $\mathcal{B}$ with P fixed at (x, y), must lie close to (x, y). Such a feature obviously has non-empty intersection with the the axis-parallel region C with (x, y) as its center and side length $2d_{\mathcal{B}}$. The (simpler) approach we use this time is slightly different from the one we used in the preceding two steps: we do not need the dynamic aspects of the data structure now, since we query with $\mathcal{O}(n)$ squares with equal side lengths $2d_{\mathcal{B}}$. We build the data structure for all obstacles in $\mathcal{O}(n \log^2 n)$ time, and then query for each critical curve section β with the squares centered at a point $(x, y) \in left(\beta)$ and a point $(x', y') \in right(\beta)$. The query time is again $\mathcal{O}(\log^2 n)$. Both queries yield a constant size set of obstacles by Lemma 5.2. From these sets we can obviously compute $\sigma(left(\beta))$ and $\sigma(right(\beta))$ in constant time, and therefore also the connectivity graph nodes defined by β. Finding the adjacencies of these nodes takes constant time as well. Repeating the process for all critical curve sections β results in $\mathcal{O}(n \log^2 n)$ computation time for finding all connectivity graph nodes. Additional $\mathcal{O}(n)$ computation time for circularly linking the nodes corresponding to the same cell (using a computed ordering of all curve sections incident at a single intersection point), guarantees that all connectivity graph edges are found in $\mathcal{O}(n \log^2 n)$.

Theorem 5.8 *Schwartz and Sharir's algorithm for motion planning of a ladder amidst polygonal obstacles can be adapted to run in $\mathcal{O}(n \log^2 n)$ for k-fat obstacles.*

The description of the algorithm given above is not complete in the sense that it makes implementation of the algorithm trivial. A few minor details are not explained, e.g. finding a point (x, y) in a region $left(\beta)$. These details, however, are not too difficult to fill in and do certainly not influence the time complexity of the algorithm.

5.3 A polygonal robot

In the preceding two subsections we have restricted our attention to a ladder moving amidst polygonal obstacles. Schwartz and Sharir's paper [7], however, also gives an algorithm for a polygonal robot.

The algorithm for a polygonal robot is similar to the algorithm for a ladder. The only difference concerns the definition of the critical curves. There are more and different types of critical curves in the polygonal case. Although the critical curves are different, the basic properties of these curves remain valid: features that are involved in the definition of a single critical curve are less than $d_{\mathcal{B}}$ apart, and each point on a critical curve is less than $d_{\mathcal{B}}$ away from the features that define it. Note that $d_{\mathcal{B}}$ is now the diameter of the robot $\mathcal{B}$. The validity of these properties allows us to use a similar proof strategy and a similar approach for an algorithm in the case of a polygonal robot, resulting in the same complexity for the cell decomposition and for the motion planning algorithm.

For the reasons mentioned above, we do not give the proof for the polygonal robot case here. The reader is referred to [11] for the details of the polygonal case.

6 Conclusion

In this paper we have shown that, under some realistic assumptions, the complexity of the free space of a robot moving amidst k-fat obstacles is linear. The combinatorial result implies better bounds for motion planning algorithms with a running time that is dependent on the complexity of the free space, like the algorithm by Sifrony and Sharir [9] for planning the motion of a ladder in R^2. Another algorithm for planning the motion of a ladder in the plane is the well-known $\mathcal{O}(n^5)$ algorithm by Schwartz and Sharir [7]. It can be proven that the complexity of their cell decomposition and the connectivity graph of this cell decomposition is linear in the number of obstacle edges and corners. Although the complexity of Schwartz and Sharir's algorithm is not fully determined by the complexity of the free space we have shown that we can modify the algorithm so that it computes the $\mathcal{O}(n)$ complexity connectivity graph in $\mathcal{O}(n \log^2 n)$ time for a ladder or a polygon moving amidst k-fat obstacles.

We expect that many more exact motion planning algorithms benefit from a low complexity of the free space (or can be adapted to do so). We are currently investigating the consequences of fatness for the generalized version of Schwartz and Sharir's cell decomposition algorithm, presented in a different paper by the same authors [8]. This algorithm provides solutions to more complex motion planning problems.

References

[1] 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, *Proc. 6th ACM Symp. on Computational Geometry* (1990), pp. 281-289.

[2] J.M. Bañon, Implementation and Extension of the Ladder Algorithm, *Proc. IEEE Int. Conf. on Robotics and Automation*, Cincinnati OH (1990), pp. 1548-1553.

[3] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston (1991).

[4] J. Matoušek, N. Miller J. Pach, M. Sharir, S. Sifrony, and E. Welzl, Fat Triangles Determine Linearly Many Holes, *Proc. 32nd IEEE Symp. on Foundations of Computer Science* (1991), pp. 49-58.

[5] M.H. Overmars, Range searching in a set of line segments, *Proc. 1st ACM Symp. on Computational Geometry* (1985), pp. 177-185.

[6] M.H. Overmars, Point Location in Fat Subdivisions, *Inform. Proc. Lett.* (1992), pp. 261-265.

[7] J.T. Schwartz and M. Sharir, On the Piano Movers' Problem: I. The Case of a Two-Dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers, *Comm. Pure Appl. Math.* **36** (1983), pp. 345-398.

[8] J.T. Schwartz and M. Sharir, On the Piano Movers' Problem: II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds, *Adv. in Applied Mathematics* **4** (1983), pp. 298-351.

[9] 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.

[10] A.F. van der Stappen, D. Halperin, and M.H. Overmars, The Complexity of the Free Space for a Robot Moving Amidst Fat Obstacles, Tech. Rep. RUU-CS-92-05, Dept. of Computer Science, Utrecht University (1992).

[11] A.F. van der Stappen, D. Halperin, and M.H. Overmars, New Efficient Algorithms for Motion Planning Amidst Fat Obstacles, to appear.