The Complexity of the Free Space for Motion Planning amidst Fat Obstacles

A. Frank van der Stappen

RUU-CS-92-31
October 1992

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

The Complexity of the Free Space for Motion Planning amidst Fat Obstacles

A. Frank van der Stappen

Technical Report RUU-CS-92-31
October 1992

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

ISSN: 0024-3275

The Complexity of the Free Space for Motion Planning amidst Fat Obstacles *

A. Frank van der Stappen

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

Abstract

The complexity of 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, resulting in bad worst-case time bounds for motion planning algorithms. In practice, the complexity of the free space tends to be much smaller than the worst-case complexity. Motion planning algorithms with a running time that is determined by the complexity of the free space therefore become feasible in practical situations. We show that, under some realistic assumptions, the complexity of the free space of a robot with any fixed number of degrees of freedom moving around in a d-dimensional Euclidean workspace with fat obstacles is linear in the number of obstacles. The complexity results lead to highly efficient algorithms for motion planning amidst fat obstacles.

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. A user of such a robot is only responsible for specifying the tasks; he is not bothered with how these tasks are executed.

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.* (An overview of the state-of-the-art in robot motion planning is given in [8].) In this paper, we consider the following version of the general motion planning problem.

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

Given a robot $\mathcal{B}$ moving amidst a collection of obstacles $\mathcal{E}$, and an initial placement Z_0 and a desired 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.

In the next section we present a mathematical formulation of the motion planning problem. This commonly used mathematical formulation is referred to as the configuration space formulation of the motion planning problem [8]. In this formulation, a placement of the robot $\mathcal{B}$ is mapped to a point in an appropriate space. The obstacles map to point sets in this space. The mapping transforms the problem of planning the motion of a dimensioned object $\mathcal{B}$ into the problem of planning the motion of a single point in the, usually higher-dimensional, appropriate space. In Section 3 we discuss the commonly used approaches for solving the motion planning problem. In Section 4 we introduce the notion of fatness for shapes. In Sections 5 and 6 we show that the complexity of the solution space for the motion planning problem is low if the obstacles that the robot is moving amidst are fat. Finally, in Section 7 it is shown that fatness reduces the time complexity of the well-known algorithm by Schwartz and Sharir for planning the motion of a ladder amidst polygonal obstacles in a two-dimensional workspace from $\mathcal{O}(n^5)$ to $\mathcal{O}(n^2)$. The algorithm is then modified so that its running time is further reduced to $\mathcal{O}(n \log^2 n)$.

2 Configuration space formulation of the motion planning problem

A robot $\mathcal{B}$ moves around in a *workspace*, or *physical space*, W. This workspace W usually corresponds to the Euclidean space of dimension two or three (R^2 or R^3), since these are the most interesting cases from a practical point of view.

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*.

Generally, the *robot* $\mathcal{B}$ is an m-tuple of rigid bodies. A rigid body is a compact connected subset of W. A *rigid robot* is a robot consisting of a single rigid body ($m = 1$). A *nonrigid robot* consists of more than one rigid body ($m > 1$). The motions of the m different components of a nonrigid robot are usually not independent from each other. In most practical situations the components are in some way attached to each other. These couplings of the components constrain the relative placements and motions of each of the components. Figure 1 shows four examples of robots. The robots in Figure 1(a),(b) live in a two-dimensional Euclidean workspace (R^2); the robots in Figure 1(c),(d) move in three-dimensional Euclidean space (R^3). Robots 1(a),(d) are examples of nonrigid robots, the other two robots are rigid (assuming that the plane does not have movable parts). Both nonrigid robots are examples of so-called *robot arms*. A robot arm is a set of rigid bodies, or *links*, that are attached to each other in a chain. Two links are attached to each other by *joints*. A joint can be either *revolute* (rotating) or *prismatic* (sliding). Robot 1(a) consists of five links, three revolute joints, and one prismatic joint. The three-dimensional

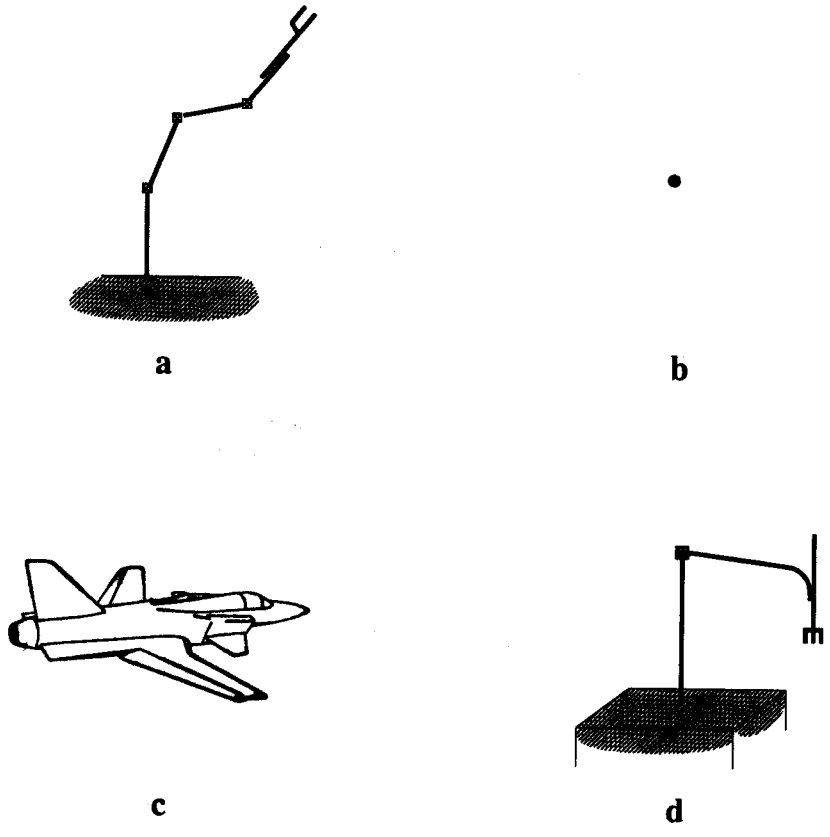

Figure 1: Examples of robots: robot arm (a) is a 4-DOF robot in R^2, point robot (b) is a 2-DOF robot in R^2, plane (c) is a 6-DOF robot in R^3, and robot arm (d) is a 2-DOF robot in R^3.

robot 1(d) consists of three links, one revolute joint, and one prismatic joint. Many known motion planning algorithms only consider (a class of) rigid robots.

In most situations it is not sufficient to characterize a placement of the robot in the workspace, by the position of some chosen reference point on the robot in the workspace. Consider for example the simple situation where a line segment PQ moves around in R^2. Characterizing a placement of PQ by the position of, for example, the endpoint P is not sufficient, because then robot PQ can still be placed in infinitely many ways. (Endpoint Q can be chosen on a full circle in R^2). If we add another parameter θ giving the angle between PQ and, say, the positive x-axis, then the triple (x, y, θ) determines one single placement of the robot PQ. Instead of two parameters x and y, we need three parameters x, y, and θ in the line segment case to characterize a placement. Hence, a placement for the robot PQ is mapped to a point in $R^2 \times [0, 2\pi)$. Such a space C of parametric representations of robot placements is called *configuration space*. The parameters that are needed to specify a robot placement are referred to as the *degrees of freedom* (DOF) of

the robot. The number of degrees of freedom of a robot determines the dimension of its configuration space. A point Z in the configuration space C is a *placement*. A placement $Z \in C$ of the robot $\mathcal{B}$ corresponds to a set of points (covered by $\mathcal{B}$) in the workspace W. This point set is denoted by $\mathcal{B}[Z]$.

Let us reconsider the example robots in Figure 1 in order to see a few examples of configuration spaces.

(a) The two-dimensional robot arm has four degrees of freedom. Assuming that the two lower revolute joints cover the full angular sector $[0, 2\pi)$, the upper revolute joint can establish all angles in the range $[\pi, 2\pi)$, and the length of the upper (fork) link is 2, then a suitable configuration space is $C = [0, 2\pi)^2 \times [0, \pi) \times [0, 2]$. A placement $Z = (\alpha, \beta, \gamma, \ell)$ gives the angle α between the first and second link, the angle β between the second and third link, the angle between the third and fourth link, and the length ℓ of the overlap of the fourth and fifth link; $Z = (\frac{5\pi}{6}, \frac{2\pi}{3}, \frac{4\pi}{3}, 1)$ is a valid placement.

(b) The point robot has two degrees of freedom, its x-coordinate and its y-coordinate. A suitable configuration space is $C = R^2$.

(c) The plane has six degrees of freedom. First we fix the back end of the plane's body as its reference point. The position of this reference point is a point $(x, y, z) \in R^3$. The front end of the plane can now be chosen on a full sphere in R^3. Like specifying a point on earth this can be done by a point in $[0, 2\pi) \times [-\frac{\pi}{2}, \frac{\pi}{2}]$. Finally, the plane can fully rotate around its own length axis. A suitable configuration space for the plane is therefore $C = R^3 \times [0, 2\pi) \times [-\frac{\pi}{2}, \frac{\pi}{2}] \times [0, 2\pi)$.

(d) The three-dimensional robot arm lives in a three-dimensional Euclidean workspace but two parameters are sufficient to specify a placement of the arm. We can choose $C = [0, 2\pi) \times [0, L]$, where L is the length of the extreme link.

A point Z in the configuration space C can correspond either to a placement of the robot $\mathcal{B}$ in the workspace W in which it intersects no obstacle, $\mathcal{B}[Z] \cap \mathcal{E} = \emptyset$, or to a placement of $\mathcal{B}$ in W in which is has non-empty intersection with the obstacle set $\mathcal{E}$. The first type of placement is called a *free placement*. The *free space* FP is the set of all free placements of the robot $\mathcal{B}$, hence

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

Sometimes it is more appropriate to allow the robot to move while being in contact with an obstacle. Then, the intersection of the robot with the obstacle set $\mathcal{E}$ is non-empty but the intersection of $\mathcal{B}$ with the interior $int(\mathcal{E})$ of $\mathcal{E}$ is empty. A placement Z such that $\mathcal{B}[Z] \cap int(\mathcal{E}) = \emptyset$ is called a *semi-free placement*. The *semi-free space* SFP is the set of all semi-free placements of the robot $\mathcal{B}$, hence

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

A *collision-free path* or *collision-free motion*, or path or motion for short, 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

$$\tau(0) = Z_0 \quad \text{and} \quad \tau(1) = Z_1.$$

Semi-free motion can be allowed by changing the range of the map τ into SFP. Hence, 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. In the next sections we will discuss the complexity of the free space in more detail.

3 Approaches to solving the motion planning problem

Motion planning algorithms are either approximate or exact. The main difference between both classes is that approximate motion planning methods may fail to find a collision-free path for the robot even if such a path exists, whereas the exact motion planning algorithms always find a path if one exists. Another important characteristic of approximate methods is that, in general, the efficiency of such methods is not related to the complexity of the free space. Approximate methods can behave very well in situations with high free space complexity but also very bad in situations with a low complexity free space. We restrict ourselves to exact motion planning methods, since these methods can benefit from the reduced free space complexities that we will prove for situations satisfying some realistic assumptions.

The *cell decomposition* approach (see e.g. [2, 6, 7, 9, 15]) is an exact motion planning technique that partitions the space 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 vertex in the *connectivity graph* CG. Two vertices 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 the cells C_0 and C_1 in which the placements Z_0 and Z_1 lie and determine a path in CG between the vertices corresponding to the cells C_0 and C_1, or report that no such path exists. Next, the resulting sequence of cells and the crossing rules for each pair of consequent cells are used to find a path for the robot $\mathcal{B}$ from Z_0 to Z_1.

Applications of the cell decomposition technique include an $\mathcal{O}(n^5)$ algorithm by Schwartz and Sharir [15] for planning the motion of a non-convex polygonal robot $\mathcal{B}$ moving amidst polygonal obstacles $\mathcal{E}$ in the plane (with a total number of n edges). Leven and Sharir [9]

present a more efficient version of the algorithm by Schwartz and Sharir for the special case of a ladder moving amidst polygonal obstacles. Their algorithm runs in time $\mathcal{O}(n^2 \log n)$. Other results for motion planning in the plane are given by Kedem and Sharir [7] and Avnaim, Boissonnat, and Faverjon [2]. Both algorithms have high worst-case time bounds: they run in roughly quadratic and cubic time respectively. In practical situations, some of the algorithms show better performance though.

A different exact approach to motion planning is the *retraction method* (see e.g. [11, 12, 16]). In this approach we define a lower dimensional subspace FP′ of FP and show that there exists a retraction function Im from FP to FP′. The retraction function Im maps each placement in FP onto a placement in FP′. Hence, motion planning in FP is reduced to motion planning in FP′. By repeated retractions we obtain a one-dimensional network N $\subseteq$ FP. Motion planning is again reduced to graph searching if we represent the one-dimensional network N as a graph.

Ó'Dúnlaing and Yap [11] and Ó'Dúnlaing, Sharir, and Yap [12] present algorithms for planning the motion of a disc and a ladder, based on retractions onto Voronoi diagrams. The algorithms run in time $\mathcal{O}(n \log n)$ and $\mathcal{O}(n^2 \log n \log^* n)$[1] respectively. Sifrony and Sharir [16] apply a variant of the retraction technique. They use a retraction that maps placements in FP onto particular vertices on the boundary of FP. The resulting algorithm runs in $\mathcal{O}(K \log n)$, where K is the number of feature pairs that are less than the length of the ladder apart. The complexity of this algorithm is determined by the complexity of the free space. Hence, not too much time is spent on other things than producing a description of the free space. Apparently, algorithms that have this property are close to optimal, because algorithms that run in less time than the time needed to produce an efficient representation of the free space can only be obtained in special cases where it is sufficient to compute only part (e.g. a single cell) of the free space. So, we must strive for algorithms with a complexity that is determined by the complexity of the free space.

As stated above, the complexity of a cell decomposition or a retraction network highly depends on the complexity of the free space FP. The complexity of the free space, as we will see in the next section, is determined by the number of multiple contacts of the robot $\mathcal{B}$ and the obstacles. A multiple contact of the robot $\mathcal{B}$ is a placement in which it touches more than one obstacle feature (e.g. edge, corner). Unfortunately the number of multiple contacts, and, hence, the complexity of the free space, 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 the configuration space) and the number of robot features is bounded by some constant, then this complexity can be $\Omega(n^f)$. So, theoretically, these cell decomposition and retraction techniques are very expensive. Fortunately, in many practical situations the complexity of FP is much smaller and, hence, these methods might become feasible [3] (assuming that the algorithms are sensitive to the size of FP, like e.g. the algorithm of Sifrony and Sharir [16]). 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.

[1] $\log^* n = \min\{\, i \geq 0 \mid \log^{(i)} n \leq 1 \,\}$, where $\log^{(i)}$ stands for the logarithm function applied i times in succession [4].

4 The free space complexity and the number of 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 the configuration space. The set SFP is the closure of the set FP, so the difference BFP = SFP - FP of these sets is the boundary of the set of free placements. By the definitions of FP and SFP we obtain:

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

Hence, the boundary BFP of the free space is the set of placements in which the robot only touches the obstacles and not really intersects them. We examine the set of placements in which the robot touches, or, is in contact with, one or more obstacles, since this set 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 (vertex, edge, face) is in contact with a specific obstacle feature is an $(f-1)$-dimensional subset, or hypersurface, in the f-dimensional configuration space. 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 n respectively, then the total number of hypersurfaces equals $\mathcal{O}(n)$.

The contact hypersurfaces described above can obviously intersect each other. A point on the intersection of two hypersurfaces corresponds to a double contact of the robot $\mathcal{B}$. The intersection of two hypersurfaces is an $(f-2)$-dimensional subset in the configuration space. All points in this subset correspond to placements in which the same two contacts exist. Any pair of hypersurfaces can theoretically be involved in such an intersection, so the total number of intersections is $\mathcal{O}(n^2)$. Moreover, a point on the intersection of j hypersurfaces corresponds to a j-fold contact of the robot with the obstacles. Such a j-fold contact defines a $(f-j)$-dimensional subset of the configuration space. The number of j-fold contacts is $\mathcal{O}(n^j)$. An f-fold contact defines a point in the configuration space. The number of f-fold contacts is $\mathcal{O}(n^f)$. 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 the boundary of the free space, and, hence, of the free space 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 the intersection of each multiple of hypersurfaces consists of only a constant number of connected sets. The latter requirement will generally be satisfied. Theorem 4.1 states this result on the relation between the complexity of the free space and the number of multiple contacts.

Theorem 4.1 *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 $\mathcal{B}$.*

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 the robot, clearly result in less multiple contacts for $\mathcal{B}$ than obstacles that are cluttered. Similarly, obstacles that have no long and skinny parts will induce less multiple contacts than obstacles that do have such parts. We may therefore expect that the actual complexity of the free space will remain far below the worst case complexity in those situations.

An obstacle that has no long and skinny parts is called *fat*. This informal description of fatness can be formalized in many different ways (see e.g. [10] and [1]). Our (slightly different) definition (see [17] and the next section) results in a free space with a complexity that is linear in the total obstacle complexity for a constant complexity robot moving amidst fat obstacles. The proof of the result is based on the following idea: we consider the smallest obstacle and prove a constant upper bound on the number of larger obstacles that lie close enough to this smallest obstacle so that both can be involved in a single multiple contact, and repeat this argument for every next smallest obstacle. Using this idea, we obtain a linear number of multiple contacts.

The importance of the linear complexity result becomes clear if we reconsider the $\mathcal{O}(K \log n)$-algorithm of Sifrony and Sharir [16]. Our result shows that the number of feature pairs that is less than the length of the ladder apart is linear in case of fat obstacles. As a consequence, the algorithm then runs in $\mathcal{O}(n \log n)$ time, whereas it might take $\mathcal{O}(n^2 \log n)$ time for non-fat obstacles.

5 Fat obstacles

Fatness turns out to be a very interesting property in computational geometry. Alt *et al.* [1] and Matoušek *et al.* [10] show that the upper bounds on the combinatorial complexity of the union of certain geometric figures are lower if these figures are fat. The union of geometric figures plays a role in many computational geometry applications. Fatness can lead to efficient algorithms. Overmars [14] shows that point location queries in fat subdivisions (no cell has long and skinny parts) in d-dimensional space can be performed in a simple way in $\mathcal{O}(\log^{d-1} n)$ time with a data structure that uses $\mathcal{O}(n \log^{d-1} n)$ storage.

Our definition of fatness [17] in a d-dimensional Euclidean workspace involves d-dimensional closed spherical regions centered at some arbitrary point in an obstacle E. The closed spherical region with radius r centered at m will be denoted by $S_{m,r}$, so

$$S_{m,r} = \{x \in R^d | d(x,m) \leq r\};$$

the boundary of this region will be denoted by $\partial S_{m,r}$, so

$$\partial S_{m,r} = \{x \in R^d | d(x,m) = r\}.$$

Spherical regions centered at a point m with a boundary that has non-empty intersection with an obstacle E play a central role in our notion of fatness. Therefore, the following definition is useful.

Definition 5.1 $[U_{m,E}, U_E]$
Let $m \in R^d$ and let $E \subseteq R^d$ be an obstacle. The set $U_{m,E}$ is defined as:

$$U_{m,E} = \{S_{m,r} \subseteq R^d \mid \partial S_{m,r} \cap E \neq \emptyset\}$$

The set U_E is defined as:

$$U_E = \bigcup_{m \in E} U_{m,E}$$

So, U_E is the set of all spherical regions with center inside E that do not fully contain E. Figure 2 shows a two-dimensional example of a circular region S_0 that does not belong to U_E and a circular region S_1 that does belong to U_E. Region S_0 does not belong to U_E because the obstacle E lies entirely inside it. Region S_1 is an element of U_E since its boundary has non-empty intersection with the obstacle E.

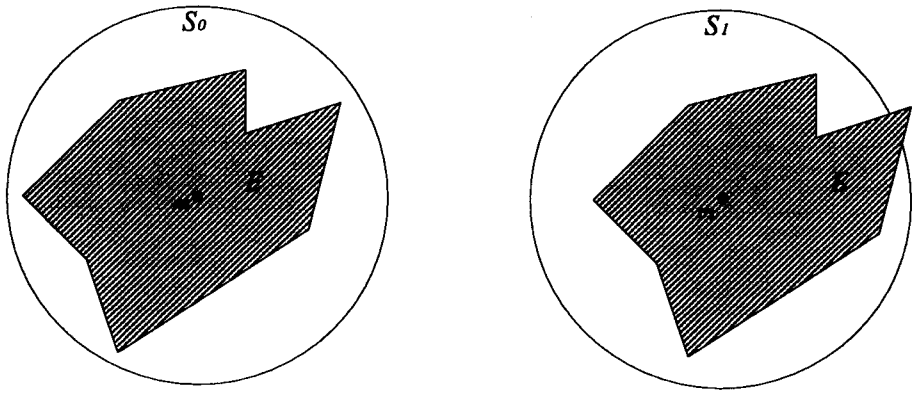

Figure 2: Illustration of the definition of U_E: $S_0 \notin U_E$ and $S_1 \in U_E$.

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. This number is a measure for the actual fatness of the obstacle. If the value of 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 5.2 [k-**fatness**]
Let $E \subseteq R^d$ be an obstacle and let k be a positive constant. The obstacle E is k-fat if:

$$\forall S \in U_E \qquad k \cdot volume(E \cap S) \geq volume(S).$$

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 the obstacle E is at least a $\frac{1}{k}^{\text{th}}$ of S. The choice for spherical regions in the definition of fatness is rather arbitrary. In fact we could have used any compact region, e.g. hypercubic regions, regions bounded by simplices etc. Any k-fat obstacle according to one definition is easily seen to be k'-fat according to another definition for some k' that is only a constant multiple of k. Note that there is a straightforward property that an obstacle that is k-fat is also k'-fat for $k' \geq k$.

No lower bound on the value of k is given in the definition of fatness. In fact the lower bound on k differs from dimension to dimension. There are for example no 1-fat obstacles at all; there can be 5-fat obstacles in a two-dimensional workspace but 5-fat obstacles in a three-dimensional workspace do not exist. This is inherent to the definition of k-fatness. Suppose we have a k-fat obstacle E with diameter δ. The volume of this obstacle is bounded from above by the volume of a hypersphere with diameter δ (or radius $\delta/2$). The diameter of E is δ, so there is a pair of points on the boundary of E that is a distance δ apart; let $m, m' \in E$ be these two points. The spherical region $S_{m,\delta}$ is an element of U_E since $m' \in \partial S_{m,\delta}$ and $m' \in E$. (Similarly, spherical region $S_{m',\delta}$ is an element of U_E.) Hence, the set U_E contains an element S with radius δ. We know that $volume(E \cap S) \leq volume(E) \leq C_d \cdot (\delta/2)^d$ and $volume(S) = C_d \cdot \delta^d$, where C_d is the dimension-dependent multiplier in the volume formulae for hyperspheres [2]. Combination with Definition 5.2 (E is k-fat and $S \in U_E$) yields $k \geq 2^d$. The boundary value 2^d-fatness is only obtained for spherical obstacles; spherical obstacles have maximal fatness.

The definition of k-fatness given in Definition 5.2 has a very "local" character: a certain portion of the proximity of every point in the obstacle must be covered by the obstacle too. As stated before, this locality prohibits obstacles with infinitesimally thin protuberances, even if these protuberances are extremely short. A huge spherical obstacle with a very short line segment sticking out of its boundary will not be k-fat for any value of k. A more natural way to define fatness might be the more "global" type of fatness given in Definition 5.3. For convenience, we will refer to it as *thickness*. Here, we only compare the volume of the entire obstacle to the volume of its minimal (volume) enclosing hypersphere: the volume of the obstacle should be at least a certain portion of the minimal enclosing hypersphere of the obstacle. This more liberal definition allows obstacles with small protuberances. If E is an obstacle then we denote the minimal enclosing hypersphere of E by MES_E.

Definition 5.3 [k-thickness]
Let $E \subseteq R^d$ be an obstacle and let $k \geq 1$ be a constant. The obstacle E is k-thick if:

$$k \cdot volume(E) \geq volume(MES_E).$$

The definition of k-thickness involves just one hypersphere instead of infinitely many. Note that not necessarily $MES_E \in U_E$: the minimal enclosing hypersphere of an obstacle can have its center outside the obstacle. Again we have the straightforward property that an

[2] For even dimension $C_d = C_{2n} = \frac{\pi^n}{n!}$. For odd dimension $C_d = C_{2n+1} = \frac{2(2\pi)^n}{(2n+1)!!}$. See, e.g., [5, Section 394].

obstacle that is k-thick is also k'-thick for $k' \geq k$. Spherical obstacles are 1-thick, because the minimal enclosing hyperspheres of such obstacles are the obstacles themselves.

Unfortunately, the notion of k-thickness does not result in low complexities of the free space. Hence, the definition is too general for our purposes. We could restrict ourselves to convex obstacles but, as we will see below in Theorem 5.4, in that case thickness is equivalent to fatness. Therefore, we have chosen to use the definition of fatness stated in Definition 5.2 because it also allows for non-convex obstacles.

Theorem 5.4 *Let $E \subseteq R^d$ be a convex obstacle. Then*

$$E \text{ is } k\text{-fat} \equiv E \text{ is } k'\text{-thick},$$

where k is only a constant multiple of k'.

We refer to [17] for the proof of this theorem.

A consequence of Theorem 5.4 is that the complexity results that we prove for convex obstacles that are k-fat also hold for convex obstacles that are k-thick. In the rest of this paper we will only consider fatness, not thickness.

Let us consider some examples of fat shapes. Spherical obstacles are obviously fat, because we have shown earlier that a d-sphere has maximal achievable 2^d-fatness. Other shapes that are fat (in two-dimensional space) include squares, rectangles with bounded aspect ratio, and triangles with angle restriction.

In [1] an approximate motion planning algorithm is given for a rectangular robot. The complexity of the algorithm decreases as the ratio of the rectangle's sides gets closer to 1. It is obvious that a (non-degenerate) rectangle is fat according to our definition. The actual fatness obviously depends, like the complexity of the algorithm, on the ratio of the rectangle's sides. The value of k will increase as the rectangle becomes "narrow". The largest spherical region in U_E (centered at a rectangle corner and enclosing the rectangle) has radius $\sqrt{a^2+b^2}$, so a rectangle with sides a and b is $\frac{\pi(a^2+b^2)}{ab}$-fat. Clearly, we get maximal fatness for squares and no fatness if either $a = 0$ or $b = 0$.

In [10] a different notion of fatness is introduced for triangles by imposing a restriction on the angles in the triangle. There, a triangle is called δ-fat if each of its three internal angles is at least δ. A δ-fat triangle is also fat according to our definition. Assume that we are given a δ-fat triangle with a longest edge e. The triangle has minimal area if the other two angles have magnitudes δ and $\pi - 2\delta$. This minimal area is $\frac{1}{4}|e|^2 \tan\delta$. The largest spherical region in U_E (centered at one of the end-points of e and enclosing the triangle) has radius $|e|$. Therefore, each δ-fat triangle is $\frac{4\pi}{\tan\delta}$-fat according to our fatness definition. Maximal fatness is obviously obtained for equilateral triangles and there is no fatness if $\delta = 0$. The latter triangle will also be non-fat in [10].

Many other classes of shapes are fat. In three-dimensional space we can think of cubes, boxes with bounded aspect ratio's in each of their bounding rectangular faces, equilateral tetrahedra etc.

6 Multiple contacts for a robot

Let us now return to determining the complexity of the free space for some motion planning problem involving fat obstacles. The actual complexity of the free space depends on the number of intersections of hypersurfaces that bound the free space. Such a hypersurface is a set of placements of the robot $\mathcal{B}$ in which a certain feature of $\mathcal{B}$ is in contact with a certain feature of the boundary of $\mathcal{E}$. Each hypersurface can intersect any other hypersurface. Intersections of hypersurfaces correspond to multiple contacts of the robot $\mathcal{B}$ with the boundary of $\mathcal{E}$. The intersections define faces on the hypersurfaces. The set of faces forms a description of the free space FP. If the number of possible multiple contacts of the robot $\mathcal{B}$ is low then the complexity of the free space is also low.

We consider the situation where a robot $\mathcal{B}$ moves amidst k-fat obstacles $E \subseteq \mathcal{E}$. The robot $\mathcal{B}$ is *not* required to be fat. We assume that the obstacles are not infinitesimally small. Let the diameter of the smallest minimal enclosing hypersphere of any obstacle be d_{min}. The robot $\mathcal{B}$ is assumed to be relatively small compared to the obstacles E: the diameter of $\mathcal{B}$ is at most $b \cdot d_{min}$, where b is some positive constant. Both assumptions are quite realistic. The assumption on the size of the obstacles basically rules out point obstacles, whereas the assumption on the size of the robot forbids unboundedly large robots. If we do not make the second assumption then one may choose a very large robot that would make the obstacles into "point" obstacles relative to the robot. In practical situations, both assumptions will be satisfied.

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 and 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 $\mathcal{O}(n^f)$, where f is the dimension of the configuration space. The dimension of the configuration space equals the number of degrees of freedom of the robot. Each obstacle $E \subseteq \mathcal{E}$ is assumed to have only a constant number of features, so the number of obstacles is $\Omega(n)$.

The assumptions in the previous two paragraphs are sufficient to prove that the number of multiple contacts for a robot $\mathcal{B}$ is at most linear in the number of obstacles. We summarize these assumptions below.

- The workspace of the robot $\mathcal{B}$ is the d-dimensional Euclidean space (R^d).
- The workspace of the robot $\mathcal{B}$ contains n k-fat obstacles $E \subseteq \mathcal{E} \subseteq R^d$.
- The diameter of the smallest minimal enclosing hypersphere of any obstacle $E \subseteq \mathcal{E}$ is d_{min}.
- The diameter $d_{\mathcal{B}}$ of the robot $\mathcal{B}$ is bounded: $d_{\mathcal{B}} \leq b \cdot d_{min}$ (constant $b > 0$).

- The robot $\mathcal{B}$ has constant complexity.
- Each obstacle $E \subseteq \mathcal{E}$ has constant complexity.
- The set of robot placements (hypersurface in the configuration space) in which a certain robot feature is in contact with a certain obstacle feature is algebraic and of bounded degree.

6.1 The proximity of an obstacle

In this subsection we consider the proximity of a k-fat obstacle as a first step in finding an upper bound on the number of multiple contacts for a robot $\mathcal{B}$. It is obvious that two obstacles that are far (more than the diameter of the robot) apart can not be involved in any multiple contact for $\mathcal{B}$. 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 considering the proximity of this large obstacle does not result in a constant upper bound on the number of obstacles that can participate in a multiple contact involving the large obstacle: 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 will count each multiple contact m for 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 this obstacle E. Lemma 6.1 (see [17] for the proof of this result and all other results in this section) states that the number of obstacles that lie in the proximity of the obstacle E_{min} with the smallest minimal enclosing hypersphere is bounded by a constant.

Lemma 6.1 *Let $\mathcal{E} \subseteq R^d$ be a set of k-fat obstacles. Let $E_{min} \subseteq \mathcal{E}$ be the obstacle with the smallest minimal enclosing hypersphere; the diameter of this hypersphere is assumed to be d_{min}. If the diameter $d_{\mathcal{B}}$ of the robot is at most $b \cdot d_{min}$ then the number of obstacles $E \subseteq R^d$ 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 number of obstacles that can participate in a multiple contact of the robot $\mathcal{B}$ increases when the obstacles become less fat (k increases) or when the diameter of the robot increases (b increases).

6.2 A linear number of multiple contacts for $\mathcal{B}$

The proximity result given in Lemma 6.1 is the key to successful application of the proof strategy presented in the previous subsection. Using that strategy we start with the obstacle with the smallest enclosing hypersphere and repeatedly consider the obstacle with the next smallest minimal enclosing hypersphere. For each obstacle E we count the number of multiple contacts for the robot $\mathcal{B}$ involving E and obstacles with larger minimal enclosing hyperspheres. Lemma 6.1 guarantees that we find a constant upper bound on this number for each obstacle. The resulting overall number of multiple contacts will be linear, which is stated in Theorem 6.2.

Theorem 6.2 *Let $\mathcal{E} \subseteq R^d$ be a set of n k-fat obstacles of constant complexity. The diameter of the minimal enclosing hypersphere of each obstacle $E \subseteq \mathcal{E}$ is at least d_{min}. Let $\mathcal{B}$ be a robot of constant complexity with f degrees of freedom and with diameter $d_{\mathcal{B}} \leq b \cdot d_{min}$. For each j $(2 \leq j \leq f)$, the number of j-fold contacts $N(j)$ of the robot $\mathcal{B}$ is linear in the number of obstacles: $\mathcal{O}(n)$.*

Note that the value of j in Theorem 6.2 ranges from 2 to f. The number of single contacts is also linear; a different hypersurface corresponds to each contact between a robot feature and an obstacle feature, giving a total of $\mathcal{O}(n)$ hypersurfaces since there are n obstacles and because of the constant complexity of the robot and each obstacle.

The $(f-j)$-dimensional subspace defined by a single j-fold contact is not necessarily connected. Figure 3 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 all contact hypersurfaces are of bounded degree, however, implies that the number of different connected subspaces induced by a single multiple contact is bounded by some small constant. The complexity of the free space is now solely determined by the number of multiple contacts. Variable j in Theorem 6.2 can only have $f-1$ different values, so the total number of multiple contacts is linear, and, hence, the free space has linear complexity.

Corollary 6.3 *Let $\mathcal{E} \subseteq R^d$ be a set of n k-fat obstacles of constant complexity. The diameter of the minimal enclosing hypersphere of each obstacle $E \subseteq \mathcal{E}$ is at least d_{min}. Let $\mathcal{B}$ be a robot of constant complexity with f degrees of freedom and with diameter $d_{\mathcal{B}} \leq b \cdot d_{min}$. The free space for the robot $\mathcal{B}$ moving amidst the k-fat obstacles of set $\mathcal{E}$ has linear complexity.*

7 Efficient motion planning amidst fat obstacles

In this section we show that we can use the results and techniques from the previous section to obtain a near-linear algorithm for a ladder (3-DOF) moving in the plane amidst k-fat polygonal obstacles. The algorithm is a modified version of the famous ladder algorithm

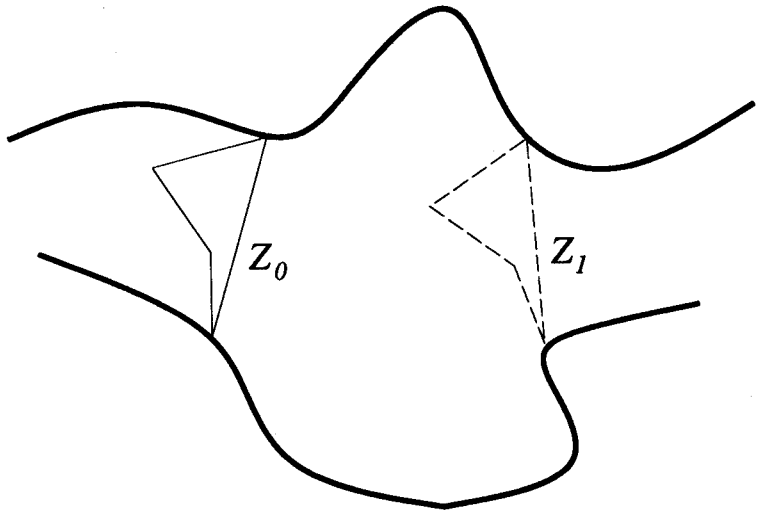

Figure 3: There is no continuous motion of the robot from Z_0 to Z_1 during which it remains in contact with both features.

by Schwartz and Sharir. The ladder algorithm runs in $\mathcal{O}(n^5)$ time in the case of arbitrary polygonal obstacles, where n is the number of obstacle edges. We show that the unmodified algorithm runs in $\mathcal{O}(n^2)$ in our case of a not too long ladder moving amidst k-fat polygonal obstacles with constant complexity. Finally, we modify the algorithm to obtain an $\mathcal{O}(n \log^2 n)$ algorithm for planning the motion of the ladder.

The method by Schwartz and Sharir decomposes R^2 into so-called *noncritical* regions, then lifts these regions into three-dimensional *cells* (in $R^2 \times [0, 2\pi)$), 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 of the obstacles for each of these steps. We emphasize that we will not give an extensive explanation of the ladder algorithm. For a more extensive explanation of the results presented in this section, the reader is referred to [18].

The noncritical regions in the robot's workspace 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 [8]. 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 4 illustrates the various critical curve types that are defined below.

- 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 the edge e is a critical curve of **type 1.** The length of the critical curve equals the length of the

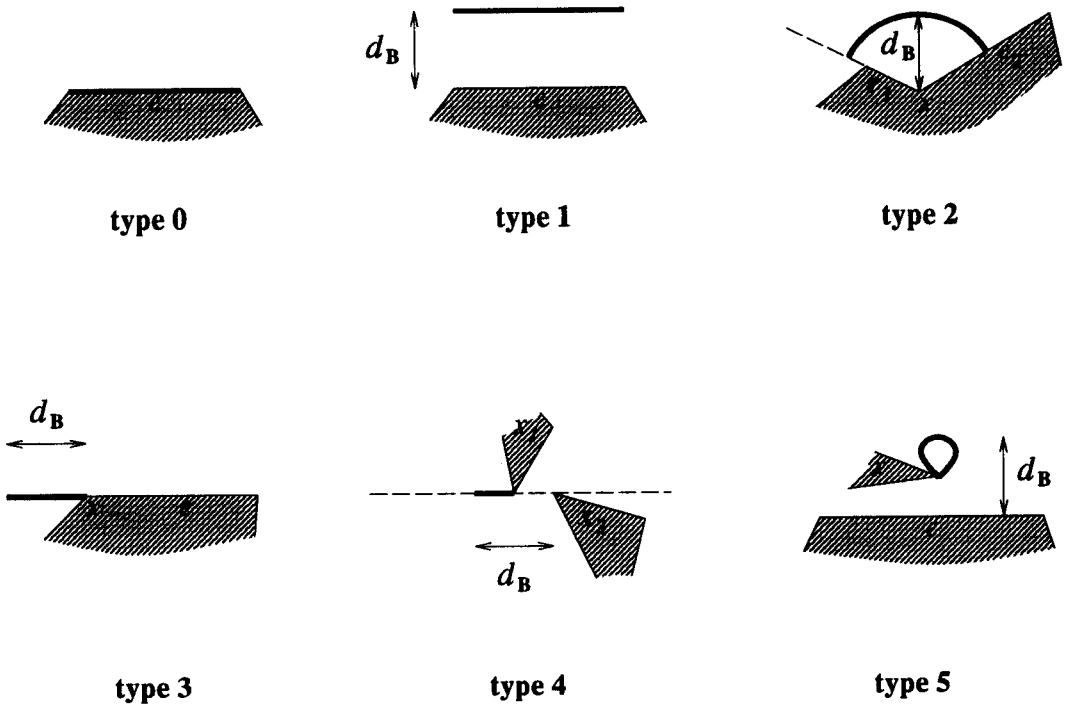

Figure 4: The six types of critical curves.

edge 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_B centered at x running between the half-line starting at x and containing the edge e_1 and the half-line starting at x and containing the edge e_2 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_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 critical curve of **type 5**. The curve is a part of a fourth degree curve known as the conchoid of Nicomedes. Note again that the distance from x to e must be less than d_B.

The (intersecting) critical curves form an arrangement in the plane. 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 the robot's workspace.

If we fix the endpoint P of the robot at a certain point (x, y) in some noncritical region R, then some orientations θ of $\mathcal{B} = PQ$ correspond to free placements whereas other orientations correspond to non-free placements. Let $\Theta_{x,y} = \{\, \theta \,|\, (x, y, \theta) \in FP \,\}$ be the set of free orientations with P fixed at (x, y). 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 $((x, y, \theta_1), (x, y, \theta_2) \in \mathrm{BFP})$. 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. We refer to [15] for a proof of this statement. In the sequel, we use the abbreviation $\sigma(R) = \sigma(x, y)$, where (x, y) is any position in the noncritical region R. Each pair of contact positions $[s_1, s_2] \in \sigma(R)$ defines a cell in the cell decomposition of the free space 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. The sections are stored sorted by outgoing tangential direction. Each critical curve section β separates two regions; we arbitrarily call one of the regions $left(\beta)$ and the other one $right(\beta)$. Next, we compute $\sigma(left(\beta))$ and $\sigma(right(\beta))$, define a connectivity graph node for each cell $[s_1, s_2]$ induced by the regions $left(\beta)$ and $right(\beta)$, 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 then circularly connected such that two nodes are linked if and only if the critical curve sections that generated these nodes are incident at the same intersection point. The orderings of the sections incident at an intersection point are used to determine these node adjacencies.

7.1 The complexity of the cell decomposition

We recall the assumption that the number of obstacle edges and corners is $\mathcal{O}(n)$. It is easy to see that each obstacle edge defines one critical curve of type 0 and one critical curve of type 1. The total number of type 0 and type 1 curves is therefore $\mathcal{O}(n)$. Each obstacle corner defines one critical curve of type 2 while each *convex* obstacle corner defines two

critical curves of type 3. Hence, the number of type 2 and type 3 curves is $\mathcal{O}(n)$.

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 curve possibly intersects any other curve, so that there can be up to $\mathcal{O}(n^4)$ critical curve 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 one important property of all critical curves. The property follows from the definitions of the critical curves.

Property 7.1 *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 6.1.

Lemma 7.2 *Let $\mathcal{E} \subseteq R^2$ be a set of k-fat polygonal obstacles. Let $E_{min} \subseteq \mathcal{E}$ be the obstacle with the smallest enclosing circle; the diameter of this circle is assumed to be d_{min}. Furthermore, let $c, c' \geq 0$ be two constants. If the diameter $d_{\mathcal{B}}$ of the robot is at most $b \cdot d_{min}$ then the number of obstacles $E \subseteq R^2$ 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.*

The number of critical curves of types 0-3 is obviously not influenced by the fatness of the obstacles: $\mathcal{O}(n)$. Now let us consider the smallest obstacle E_{min}. We count the number of type 4 and 5 curves with at least one of their defining features on 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, because the curve is defined by the positions of $\mathcal{B}$'s reference point, while $\mathcal{B}$ moves in simultaneous contact with both f_1 and f_2. 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 the sides outward by a distance $d_{\mathcal{B}}$ (see Figure 5). Lemma 7.2 with $c = 1$ and $c' = 2$ shows that the number of obstacles with 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 obstacle 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 an overall $\mathcal{O}(n)$ number of type 4 and type 5 curves. Adding all the linear numbers of different critical curves, we obtain Lemma 7.3.

Lemma 7.3 *The number of critical curves in the workspace of a ladder moving amidst k-fat obstacles is $\mathcal{O}(n)$.*

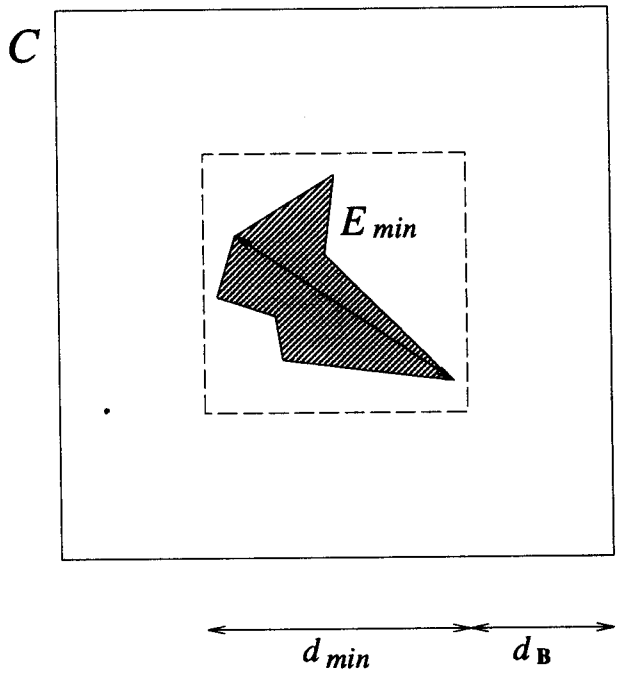

Figure 5: The square region C obtained by first taking a square region with side length d_{min} (the diameter of E_{min}) and subsequently enlarging this square region by a distance $d_{\mathcal{B}}$.

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 7.1. Let F be the set of features defining β'. Again by Property 7.1, 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 7.2 with $c = 1$ and $c' = 4$ and the constant complexity of the obstacles establish that the number of possible sets F is bounded by a constant, and, hence, the number of critical curves β' intersecting β. Since all critical curves are algebraic and of low degree, i 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 the result stated in Lemma 7.4 and Corollary 7.5.

Lemma 7.4 *The number of critical curve intersections in the workspace of a ladder moving amidst k-fat obstacles is* $\mathcal{O}(n)$.

Corollary 7.5 *The number of critical curve sections in the workspace of a ladder moving amidst k-fat obstacles is* $\mathcal{O}(n)$.

In the introductory part of this section, 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 connectivity 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 $\sigma(R)$ on its turn, consists of two different contact placements for $\mathcal{B}$ with its endpoint P fixed at some point in R. The number of cells induced by R is therefore also 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 general 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 result in $\mathcal{O}(n)$ 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) (the intersection point of both diagonals). The region C with diameter $2d_{\mathcal{B}}$ centered at $(x, y) \in R$ satisfies again the constraints of Lemma 7.2 ($c = 0, c' = 2$). 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 7.6.

Lemma 7.6 *Each noncritical region in the workspace of a robot moving amidst k-fat obstacles induces only $\mathcal{O}(1)$ cells in the configuration space.*

The result of Lemma 7.6 shows that each critical curve β adds at most twice a constant number of nodes to the connectivity graph. By Lemma 7.5, we conclude that the total number of nodes is $\mathcal{O}(n)$. The number of graph edges also turns out to be $\mathcal{O}(n)$. 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 7.6 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 7.7 *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.*

7.2 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. The first step would make the algorithm run in $\theta(n^2)$.

We have shown in the previous subsection that the number of type 4 and type 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 - such that f_1 and f_2 together define a type 4 or type 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 obviously 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. After having computed the critical curves involving E_{min}, we forget obstacle 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 uniquely defined; it is obtained by taking any enclosing square of the obstacle with diameter equal to the diameter of the obstacle and subsequently growing this enclosing square in all directions by the diameter of the robot. The region C can have any orientation, since the initial enclosing square can be chosen with any orientation. Finding a solution to the problem is simplified if we choose all square regions to have the same (axis-parallel) direction.

The axis-parallel square intersection problem can be solved using the following theorem from [13].

Theorem 7.8 *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 related to E. Our approach is the following. We order the, say n, obstacles in $\mathcal{E}$ by increasing size of their enclosing squares: $E_1 \ldots E_n$, where 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 $E_{i+1} \ldots E_n$. 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 the diameter $d_{\mathcal{B}}$ of the robot). Then we can search with the square region C_i in the data structure containing the boundary edges of the obstacles $E_{i+1} \ldots E_n$ and find the constant number of intersecting obstacles ($K = \mathcal{O}(1)$ by Lemma 7.2) 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 ($E \subseteq \mathcal{E}$) 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 guarantees that, if we repeat this procedure for every next smallest obstacle, the total number of feature pairs considered is $\mathcal{O}(n)$, and therefore equal to the number of critical curves (Lemma 7.3). The most time consuming part so far is the computation of the sets V_E, taking overall $\mathcal{O}(n \log^2 n)$ time.

After having computed the critical curves, 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 $\sigma(\mathit{left}(\beta))$ and $\sigma(\mathit{right}(\beta))$. Computing $\sigma(\mathit{left}(\beta))$ is done by taking a point $(x, y) \in \mathit{left}(\beta)$, 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 (intersection point of its diagonals) 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 induced by a point $(x, y) \in \mathit{left}(\beta)$ and a point $(x', y') \in \mathit{right}(\beta)$. The query time is again $\mathcal{O}(\log^2 n)$. Both queries yield a constant size set of features by Lemma 7.2. From these sets we can obviously compute $\sigma(\mathit{left}(\beta))$ and $\sigma(\mathit{right}(\beta))$ 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 the ordering by outgoing tangential direction of all curves incident at

a single intersection point), guarantees that all connectivity graph edges are found in $\mathcal{O}(n \log^2 n)$ time as well.

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

Combining the complexity of each of the three steps mentioned above results in an $\mathcal{O}(n \log^2 n)$ algorithm for computing the connectivity graph for the cell decomposition of the free space of a not too long robot moving amidst k-fat obstacles. The algorithm presents a considerable gain in in efficiency for fat motion planning compared to the original $\mathcal{O}(n^5)$ algorithm by Schwartz and Sharir.

8 Conclusion

In this paper we have shown that, under some realistic assumptions, the complexity of the free space of a robot $\mathcal{B}$ moving amidst k-fat obstacles is linear. The assumptions include constant complexity requirements for the robot and the obstacles, a lower bound on the size of the smallest obstacle, and an upper bound on the diameter of the robot. In most practical situations, all these constraints will be satisfied. The assumption on the size of the smallest obstacle in fact only forbids point obstacles. The upper bound on the size of the robot depends on the size of the smallest obstacle. It basically states that the robot should not be too big compared to the obstacles. The constant complexity requirement and an additional bound on the degree of the hypersurfaces defined by robot-obstacle contacts can be translated into a requirement that the boundary of the obstacles and the robot do not have a too complex shape. Both assumptions are common assumptions in motion planning.

If an obstacle does not meet the complexity assumption, the linear complexity result presented in this paper can still be applicable. First of all, it may very well be possible to partition the obstacle into a constant number of sub-obstacles, such that each sub-obstacles satisfies the complexity assumption for obstacles. After the partitioning, each sub-obstacle is treated as a separate obstacle. If the complexity is too high to apply this procedure, an adequate constant complexity outer approximation of the obstacle might be found with a volume that is not too much larger than the volume of the obstacle itself. If we can find such an outer approximation then we can do motion planning for this approximation at the cost of a relatively small reduction of the number of solutions. A constant complexity outer approximation is certainly superior to a simple approximation like a minimal enclosing hypersphere with respect to minimizing the reduction of the solution space. Similar procedures can be applied if the complexity of the robot is too high.

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 [16] for planning the motion of a ladder in two-dimensional space.

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 [15]. 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 moving amidst k-fat obstacles.

Acknowledgments

The author would like to thank Mark Overmars for useful comments on an earlier version of this paper.

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] F. Avnaim, J.D. Boissonnat, and B. Faverjon, A Practical Exact Motion Planning Algorithm for Polygonal Objects Amidst Polygonal Obstacles, Rapports de Recherche 890, INRIA, France (1988).

[3] J.M. Bañon, Implementation and Extension of the Ladder Algorithm, *Proceedings of the IEEE International Conference on Robotics and Automation*, Cincinnati OH (1990), pp. 1548-1553.

[4] T.H. Cormen, C.E. Leierson, and R.L. Rivest, *Introduction to Algorithms*, The MIT Press, Cambridge MA (1990).

[5] G.M. Fikhtengol'ts, *The Fundamentals of Mathematical Analysis, Vol. II*, English edition, Pergamon Press (1965).

[6] D. Halperin, M.H. Overmars, and M. Sharir, Efficient motion planning for an L-shaped object, *SIAM Journal on Computing* **21** (1992), pp. 1-23.

[7] K. Kedem and M. Sharir, An Efficient Motion-Planning Algorithm for a Convex Rigid Polygonal Object in Two-Dimensional Polygonal Space, *Discrete Comput. Geom.* **5** (1990), pp. 43-75.

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

[9] 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.

[10] 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.

[11] 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.

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

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

[14] M.H. Overmars, Point Location in Fat Subdivisions, Technical Report RUU-CS-91-40, Department of Computer Science, Utrecht University (1991).

[15] 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.

[16] 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.

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

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