Dynamic Motion Planning in Low Obstacle Density Environments [*]

Robert-Paul Berretty   Mark Overmars   A. Frank van der Stappen

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

Abstract

A fundamental task for an autonomous robot is to plan its own motions. Exact approaches to the solution of this motion planning problem suffer from high worst-case running times. The weak and realistic low obstacle density (L.O.D.) assumption results in linear complexity in the number of obstacles of the free space [11]. In this paper we address the dynamic version of the motion planning problem in which a robot moves among moving polygonal obstacles. The obstacles are assumed to move along constant complexity polylines, and to respect the low density property at any given time. We will show that in this situation a cell decomposition of the free space of size $O(n^2\alpha(n)\log^2 n)$ can be computed in $O(n^2\alpha(n)\log^2 n)$ time. The dynamic motion planning problem is then solved in $O(n^2\alpha(n)\log^3 n)$ time. We also show that these results are close to optimal.

Keywords: Motion planning, low obstacle density, moving obstacles, cell decomposition.

1 Introduction

Robot motion planning concerns the problem of finding a collision-free path for a robot $\mathcal{B}$ in a workspace W with a set of obstacles $\mathcal{E}$ from an initial placement Z_0 to a final placement Z_1. The parameters required to specify a placement of the robot are referred to as the degrees of freedom of the robot. The motion planning problem is often studied as a problem in the configuration space C, which is the set of parametric representations of the placements of the robot $\mathcal{B}$. The free space *FP* is the sub-space of C of placements for which the robot does not intersect any obstacle in $\mathcal{E}$. A feasible motion for the robot corresponds to a curve from Z_0 to Z_1 in *FP* (or its closure).

Motion planning is a difficult problem. In general, many instances of the robot motion planning problem are P-SPACE-complete, even if the obstacles are stationary [5]. For a constant-complexity robot moving amidst stationary obstacles polynomial time algorithms have been shown to exist. The running time is exponential in the number of degrees of freedom of the robot [7]. For an f-DOF robot, the complexity of the free space, can be as high as $\Omega(n^f)$ and the motion planning problem will, therefore, in general have a worst case running time close to $\Omega(n^f)$.

[*]Research is partially supported by the Dutch Organization for Scientific Research (N.W.O.).

We address the motion planning problem for a robot operating in an environment with moving obstacles. This problem is also referred to as the dynamic motion planning problem. In general, when the obstacles in the workspace are allowed to move, the motion planning problem becomes even more complicated. For example, Reif and Sharir [6] showed that, when obstacles in a 3-dimensional workspace are allowed to rotate, the motion planning problem is PSPACE-hard if the velocity modulus is bounded, and NP-hard otherwise. (A similar result was obtained by Sutner and Maass [8].) Canny and Reif [3] showed that dynamic motion planning for a point in the plane, with a bounded velocity modulus and an arbitrary number of convex polygonal obstacles, is NP-hard, even when the obstacles are convex and translate at constant linear velocities. They also showed that the 2-dimensional dynamic motion planning problem for a translating robot $\mathcal{B}$ with bounded velocity modulus, among polygonal obstacles $\mathcal{E}$ that translate at fixed linear velocity, can be solved using an algorithm that is polygonal in the total number of vertices of $\mathcal{B}$ and $\mathcal{E}$, if the number of obstacles is bounded. However, their algorithm takes exponential time in the number of moving obstacles.

Van der Stappen *et al.* [11] (see also [10]) showed that modelling robots in realistic workspaces has a profound influence on the complexity of solving the static motion planning problem, mainly independent of the number of degrees of freedom of the robot. They gave a description of environments with a so-called *low obstacle density* which leads to a surprising gain in efficiency for several instances of the motion planning problem. An environment has the low obstacle density property if any region in the workspace intersects a constant number of obstacles that are larger than the size of the region. (See below for a more precise definition.) Under the low obstacle density assumption, the exact motion planning problem for an f-DOF robot was efficiently solved, using the cell decomposition approach. The low obstacle density of the workspace implies a linear combinatorial complexity of the free space, even for f-DOF robots. For a robot $\mathcal{B}$ moving amidst n stationary obstacles the cell decomposition of the free space has $O(n)$ size and is computable in $O(n \log n)$ time. Vleugels [12] extended these results to multiple robots simultaneously operating in the same workspace. De Berg *et al.* [2] gave an overview of several realistic input models and gave experimental results on scenes based on real input data, which showed that the 'hidden' constant in the low obstacle density assumption was indeed low.

We demonstrate that the low obstacle density property can also be used to efficiently plan a motion for a robot $\mathcal{B}$ with f degrees of freedom moving in a 2-dimensional workspace with non-stationary obstacles. The obstacles are allowed to translate in the workspace along polyline trajectories, with a fixed speed per segment. The motion planning problem is then solved in $O(n^2\alpha(n)\log^3 n)$ time, using a cell decomposition of size $O(n^2\alpha(n)\log^2 n)$. Note that these bounds do not depend on f (assuming f is constant). We also show that this result is close to optimal, by giving an example where the robot has to perform $\Omega(n^2)$ simple motions to get from its start to its goal position.

In this paper we will first present an overview of the method used in the paper of Van der Stappen *et al.* [11]. The computation of the cell decomposition for the dynamic low obstacle density motion planning problem is treated in Sections 3 and 4; the algorithm to compute a feasible path through the cell decomposition is presented Section 5. Section 6 concludes the paper.

2 Low Obstacle Density

In this section we recall some of the definitions and results from the paper by Van der Stappen *et al.* [11] on motion planning in low density environments. The authors focus in particular on the large class of motion planning problems with configuration spaces of the form $C = W \times D$, where W is the d-dimensional workspace and D is some $(f - d)$-dimensional rest space. Let us use the *reach* of a robot as a measure for its maximum size; the reach $\rho_{\mathcal{B}}$ of $\mathcal{B}$ is defined as the maximum radius that the minimal enclosing hypersphere of the robot, centered at its reference point, can ever have (in any placement of $\mathcal{B}$). The reach of the robot is assumed to be comparable to the size of the smallest obstacle. The robot has constant complexity and moves in a workspace with constant-complexity obstacles. The workspace satisfies the static low obstacle density property which is defined as follows.

Property 2.1 *Let $\mathbb{R}^d$ be a space with a set $\mathcal{E}$ of non intersecting obstacles. Then $\mathbb{R}^d$ is said to be a* static low (obstacle) density *space if for any region $R \subset \mathbb{R}^d$ with minimal enclosing hyper-sphere radius ρ, the number of obstacles $E \in \mathcal{E}$ with minimal enclosing hyper-sphere radius at least ρ intersecting R is bounded by a constant.*

Van der Stappen *et al.* [11] showed that, under the circumstances outlined above, the complexity of the free space is linear in the number of obstacles.

The configuration space contains hyper-surfaces of the form $f_{\phi,\Phi}$, consisting of placements of the robot $\mathcal{B}$ in which a robot feature ϕ is in contact with an obstacle feature Φ. We shall denote the fact that ξ is a feature of some object or object set X by $\xi \in_f X$. The arrangement of all (constant-complexity) constraint hyper-surfaces $f_{\phi,\Phi}(\phi \in_f \mathcal{B}, \Phi \in_f \mathcal{E})$ divides the higher-dimensional configuration space into free cells and forbidden cells. Van der Stappen *et al.* [11] considered so-called *cylindrifiable* configuration spaces $C = B \times D$ which have the property that the subspace B—referred to as the *base space*—can be partitioned into constant complexity regions R satisfying

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

A partition that satisfies this constraint is called a *cylindrical partition*. In words, the lifting of the region R into the configuration space is intersected by a constant number of constraint hyper-surfaces. These hyper-surfaces subdivide the cylinder $R \times D$ into $O(1)$ constant-complexity free and forbidden cells. The cylindrical partition of B therefore almost immediately gives us a cell decomposition of the free portion *FP* of C. Theorem 2.2 states that the transformation of a cylindrical partition of the base space into a cell decomposition of the free space can be accomplished in time proportional to the size of the cylindrical partition.

Theorem 2.2 *[11] Let V be the set of regions of a cylindrical partition of a base space B and let E be the set of region adjacencies. Let the regions of B be of constant complexity. Then the cell decomposition of the free space calculated by lifting the regions R of the base partition into the configuration space consists of constant complexity subcells. Furthermore, the complexity of the decomposition and the time to compute it is $O(|V| + |E|)$.*

Note that the size of the cylindrical partition determines the size of the cell decomposition. The low obstacle density motion planning problem outlined above was shown to yield a cylindrifiable configuration space, in which the workspace W is a valid base space. Small and efficiently computable cylindrical partitions of W have led to optimal cell decompositions and thus efficient solutions to the motion planning problem (see [11] for details).

In this paper, we show that the configuration space of the dynamic version of the low obstacle density motion planning problem is cylindrifiable as well. We find a cylindrical partition of an appropriate base space that leads to an almost optimal size cell decomposition.

3 A Dynamic Base Space

3.1 Problem Statement

We now focus on the dynamic robot motion planning problem, subjected to low obstacle density. We show that the framework outlined in Section 2 can be used to plan a motion for a robot $\mathcal{B}$ with f degrees of freedom, moving in a 2-dimensional workspace with non-stationary obstacles. The obstacles translate in the workspace, and can only change speed or direction a constant number of times. We will use a cell-decomposition based on a cylindrical partition, similar to Section 2. Since dynamic motion planning is tedious to deal with, we split the problem into sub-problems. We first formally define the problem and state some useful properties of the base space for the dynamic motion planning problem. In Section 4, we construct a cylindrical decomposition, and in Section 5, we compute the actual path for the robot.

The *dynamic low obstacle density motion planning problem* is defined as follows.

- The workspace W of the robot $\mathcal{B}$ is the 2-dimensional Euclidean space $\mathbb{R}^2$ and contains a collection of n obstacles $E \in \mathcal{E}$, each moving along a polyline at constant speed per line segment.
- The robot $\mathcal{B}$ has constant complexity and its reach is bounded by $\rho_{\mathcal{B}} \leq b \cdot \rho$, where $b \geq 0$ is a constant and ρ is a lower bound on the minimal enclosing hyper-sphere radii of all obstacles $E \in \mathcal{E}$.
- Each obstacle $E \in \mathcal{E}$ is polygonal and has constant complexity.
- Any constraint hyper-surface in the configuration space corresponding to the set of robot placements in which a certain robot feature is in contact with a certain obstacle, is algebraic of bounded degree.
- The robot is placed at the initial placement Z_0 at time t_0 and has to be at the goal placement Z_1 at time t_1.
- At any time between t_0 and t_1, the workspace with obstacles satisfies the low obstacle density property.

A standard approach when dealing with moving obstacles is to augment the stationary configuration space with an extra time dimension T. In this manner, we obtain the configuration-time space. When planning the motion of our robot through the configuration-time space, we have to make sure that the path is time-monotone—the robot is not allowed to move back in time. The first objective in solving the dynamic low obstacle density motion planning problem is to obtain a cylindrical partition that consists of constant complexity regions. An appropriate choice for a base space B is the Cartesian product of the 2-dimensional workspace and time. This way, the configuration time space is of the form $CT = W \times T \times D = \mathbb{R}^2 \times \mathbb{R} \times D$ $(= \mathbb{R}^3 \times D)$, where D is some $(f-2)$-dimensional rest space.

3.2 Characteristics of the Base Space

The base space $B = W \times T$ can be considered as a 3-dimensional Euclidean space. In our dynamic motion planning setting, we only consider the work-time space slice $\mathbb{R}^2 \times [t_0, t_1]$. We first look at the situation where the obstacles move along a line in the workspace. Later, we extend the result to the polyline case.

Definition 3.1 *Let $S \subseteq W$ and let γ be a curve in $W \times T$. Then the* column $col_\gamma(S)$ *is defined by $col_\gamma(S) = \{(x, y, 0)|(x, y) \in S\} \oplus \gamma$, where $\oplus$ denotes the Minkowski sum operator.*

The column $col_\gamma(S)$ is the volume swept by S in the work-time space as its reference point O follows the curve γ. In our application, the curve γ describes the translational motion of an obstacle and is therefore time-monotone. A point (x, y, t) belongs to $col_\gamma(S)$ if and only if S covers the point (x, y) at time t.

Definition 3.2 *Let $Q_{O,\rho_\mathcal{B}}$ be a square centered at the origin, having side length $2\rho_\mathcal{B}$. Then $H(E) = E \oplus Q_{O,\rho_\mathcal{B}}$.*

The Minkowski sum $H(E)$ encloses E. No point in $H(E)$ has a distance larger than $\sqrt{2} \cdot \rho_\mathcal{B}$ to E. We denote the arrangement of the boundaries $\partial col_\gamma(H(E))$ of the grown obstacle columns by $\mathcal{A}(col \circ H)$. We will show that this arrangement is of $O(n^2)$ complexity.

Let us for a moment consider a fixed obstacle E at a fixed time t_i. We consider the boundary of the grown obstacle $H(E)$. Now, if the reference point of the robot is placed outside $H(E)$, the robot cannot collide with the obstacle. If the reference point of the robot is inside the grown obstacle, there might be configurations in which the the robot intersects the obstacle. Since both the robot and the obstacles have constant complexity, the arrangement of constraint hypersurfaces in $H(E)$ at time t_i, when lifted into the configuration space, has constant complexity as well. We exploit this observation to build a partition of the base space.

We say that an obstacle E is in the proximity of another obstacle E' if $H(E)$ and $H(E')$ intersect, hence $col_\gamma(H(E))$ and $col_{\gamma'}(H(E'))$ intersect.

Theorem 3.3 *The complexity of the arrangement $\mathcal{A}(col \circ H)$ of the boundaries of the grown obstacle columns is $O(n^2)$.*

Proof: The complexity of the arrangement is determined by the number of vertices. A vertex results from an intersection of three columns. A necessary condition for three columns to intersect is that the corresponding obstacles are less than $2\sqrt{2}\rho_{\mathcal{B}}$ apart at some moment in time. We show that the number of such triples is $O(n^2)$. We charge each such triple to a pair of obstacles. For this we choose the smallest obstacle E of the three and the one (of the remaining two) that last entered E's proximity. Assume that an obstacle E' enters the proximity of E. (Note that E' can enter E's proximity at most $O(1)$ times because E and E' have constant complexity and both move along line paths.) A third obstacle E'' involved in a triple (E, E', E'') must already be in the proximity of E at the time of arrival of E' in order to be charged to the pair (E, E'). By Property 2.1, there are only $O(1)$ larger obstacles in E's proximity at any time, so E'' is chosen from a set of $O(1)$ size. As a result, only $O(1)$ triples are charged to each of the $O(n^2)$ pairs (E, E'). Each of these $O(n^2)$ triples (E, E', E'') contribute a constant number of vertices to $\mathcal{A}(col \circ H)$ because the obstacles E, E', and E'' have constant complexity and move along line paths. Therefore, the complexity of $\mathcal{A}(col \circ H)$ is bounded by $O(n^2)$. □

It is easy to see that a 2-face of a column in the final arrangement is divided into a number of parts, of which some are non-convex. The following theorem states that the 2-faces of the arrangement $\mathcal{A}(col \circ H)$ are polygons without holes. This property turns out to be important in the sequel.

Theorem 3.4 *The faces of $\mathcal{A}(col \circ H)$ are polygonal and have no holes.*

Proof: The faces of $\mathcal{A}(col \circ H)$ are formed by the possibly intersecting faces of the columns $col_\gamma(H(E))$ $(E \in \mathcal{E})$. Since the columns are polyhedra, the arrangement $\mathcal{A}(col \circ H)$ has polygonal faces. It remains to prove that the faces do not contain holes. A face of the arrangement has a hole *iff* a column penetrates the interior of this face without intersecting its boundary. We distinguish the bottom and top faces and the side faces of the columns. The bottom and top faces of the columns, i.e. the intersections of the columns with $t = t_0$ and $t = t_1$, are the boundaries of the Minkowski sums of the obstacles at their positions at t_0 and t_1 and $Q_{O,\rho_{\mathcal{B}}}$. A grown obstacle cannot be fully contained in another grown obstacle, otherwise the obstacles would also intersect, which is not the case. Therefore, the top and bottom faces of columns are faces without holes.

The side faces of the columns are the possibly intersecting walls that connect the top and bottom faces of the columns. Assume, for a contradiction, that (a part of) some side face f of $col_\gamma(H(E))$ has a hole. There must be another column $col_{\gamma'}(H(E'))$ which intersects this face. We call the smallest time coordinate of the hole t_a, and the largest time coordinate t_b. Note that $t_0 < t_a < t_b < t_1$. Without loss of generality, we fix object E, such that its speed becomes zero, and adjust the speed of the other objects accordingly. After this transformation, we consider the 2-dimensional vertical projection onto the workspace of $col_\gamma(H(E))$ and $col_{\gamma'}(H(E')) \cap \{t_a, t_b\}$ (i.e. $H(E')$ at t_a and t_b respectively. See Figure 1). Note that E and E' are grown using the same square $Q_{0,\rho_{\mathcal{B}}}$. It is easy to see that, dependent on the location of the obstacle E with respect to the projection of f, E' intersects E at $t = t_a$ or $t = t_b$ which is impossible by assumption. So, the faces of $\mathcal{A}(col \circ H)$ are polygonal and have no holes. □

If we extend the setting to the case in which obstacles $E \in \mathcal{E}$ translate along polylines, the complexity of the arrangement $\mathcal{A}(col \circ H)$ does not increase asymptotically—in

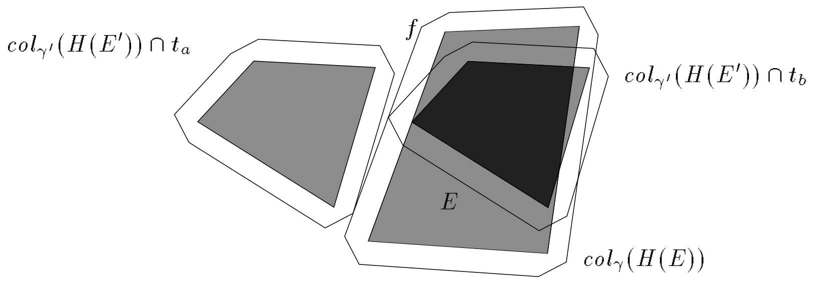
$col_{\gamma'}(H(E')) \cap t_a$
f
$col_{\gamma'}(H(E')) \cap t_b$
E
$col_{\gamma}(H(E))$

Figure 1: The 2-dimensional scene with the dark grey area depicting the intersection of two obstacles at $t = t_b$.

the proof of Theorem 3.3, the chargings to the obstacle pair (E, E') caused by obstacle E'' are, in the worst case, multiplied by a constant factor. Unfortunately, the 2-faces of $\mathcal{A}(col \circ H)$ are no longer polygons without holes. We can resolve this by adding extra faces to the arrangement. For every time t_i at which one of the obstacles changes speed, we add a plane $t = t_i$. This way, the area between two successive planes is a work-time space slice where all obstacles move in a fixed direction with a fixed speed. The arrangements on the newly introduced planes are cross sections of the work-time space. They are arrangements of possibly intersecting grown obstacle boundaries and have linear complexity because the obstacles statisfy the low obstacle density property at any time [11]. We compute a triangulation of these 2-dimensional arrangements to assure that their faces have no holes. Since we have $O(n)$ polyline vertices, the total added complexity is $O(n^2)$.

We will show that every cylinder $R \times D$, defined by a 3-cell R of the arrangement, is intersected by a constant number of constraint hyper-surfaces. We define the *coverage* of a region $R \subseteq B = W \times T$.

Definition 3.5 $Cov(R) = \{E \in \mathcal{E} | R \cap col_{\gamma}(H(E)) \neq \emptyset\}$.

In words, the coverage of a region is the set of obstacles whose columns, which are computed after growing the obstacles, intersect the region. The following result follows from the low density property and the observation that all points p in a single 3-cell of the arrangement of column boundaries lie in exactly the same collection of columns.

Lemma 3.6 *The regions R, defined by the cells of $\mathcal{A}(col \circ H)$ have $|Cov(R)| = O(1)$.*

Lemma 3.7 shows that the partition of the base space into regions R with $|Cov(R)| = O(1)$ is a cylindrical partition. The proof is very similar to the proof of Lemma 3.6 of Van der Stappen *et al.* [11] and has been omitted.

Lemma 3.7 *Let $R \subseteq B$ be such that $|Cov(R)| = O(1)$. Then*

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

The only problem is that the complexity of the cells of $\mathcal{A}(col \circ H)$ is not necessarily constant. So, we must refine the partition to create constant complexity subcells. This is discussed in Section 4.

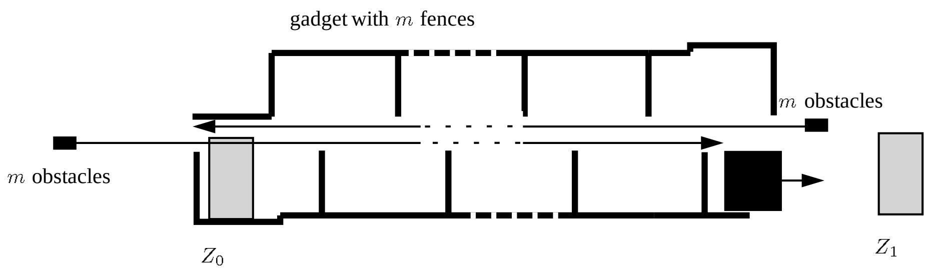

Figure 2: The quadratic lowerbound construction.

3.3 Complexity of the Free Space

In the previous subsection we showed that the work-time space of the robot can be partitioned into regions with total combinatorial complexity $O(n^2)$. Furthermore, by Lemmas 3.6 and 3.7, each region, when lifted into the configuration-time space is intersected by at most a constant number of constraint hyper-surfaces of bounded algebraic degree. Therefore, a decomposition of the configuration space into free and forbidden cells of combinatorial complexity $O(n^2)$ exists. Obviously, this $O(n^2)$ bound is an upper bound on the complexity of the free space for our dynamic motion planning setting.

Theorem 3.8 *The complexity of the free space of the dynamic low obstacle density motion planning problem is $O(n^2)$.*

We will now demonstrate that this bound is worst-case optimal, even in the situation where the robot is only allowed to translate and the obstacles move along lines. To this end, we give a problem instance with n obstacles, for which any path for the robot $\mathcal{B}$ has $\Omega(n^2)$ complexity. Consider the workspace in Figure 2. The grey rectangular robot must translate from position Z_0 to Z_1. The gadget in the middle forces the robot to make $\Omega(m)$ moves to move from left to right. It can easily be constructed from $O(m)$ stationary obstacles. The big black obstacle at the bottom right moves very slowly to the right. So it takes a long time before the robot can actually get out of the gadget to go to its goal. Now a small obstacle moves from the left to the right, through the gaps in the middle of the gadget. This forces the robot to go to the right as well. Only there can it move slightly further up to let the obstacle pass. But then a new obstacle comes from the right through the gaps, forcing the robot to move to the left of the gadget to let the obstacle pass above it. This is repeated m times after which the big obstacle is finally gone and the robot can move to its goal. The robot has to move $2m$ times through the gadget, each time making $\Omega(m)$ moves, leading to a total of $\Omega(m^2)$ moves. As $m = \Omega(n)$, the total number of moves is $\Omega(n^2)$. It is easily verified that at any moment the low obstacle density property is satisfied.

Theorem 3.9 *The complexity of the free space of the dynamic low obstacle density motion planning problem for a translating robot is $\Omega(n^2)$.*

Actually, the example shows a much stronger result. Not only does it give a bound on the complexity of the free space, but also on the complexity of a single cell in the free space and on the complexity of any dynamic motion planning algorithm.

Theorem 3.10 *The complexity of any algorithm for the dynamic low obstacle density motion planning problem (even for a translating robot) is lower bounded by $\Omega(n^2)$.*

4 Decomposing the Base Space

We still need to decompose the arrangement of columns $col_\gamma(H(E))$ $(E \in \mathcal{E})$ into constant complexity subcells. To this end, we construct a vertical decomposition of the arrangement. Since the vertical decomposition refines the cells of the arrangement, the subcells of the final decomposition still have constant-size coverage. The approach we use [1] requires that the columns in the work-time space, as described in Section 3.2, are in general position. This can be achieved by an appropriate perturbation of the vertices of the columns. Before we can calculate a vertical decomposition we have to triangulate the 2-faces of the columns. Triangulation does not increase the asymptotic complexity of the arrangement. After triangulation, the 2-faces of the arrangement might coincide, though. It is easily verified that the vertical decomposition algorithm still works with these introduced degeneracies. To bound the space we add two horizontal planes at time t_0 and t_1 (the start and goal time) and only consider the area in between. To bound the space in the x- and y-direction we also add a triangular prism far around the relevant region of the work-time space.

4.1 The Vertical Decomposition

Let $S = \{s_1, \ldots, s_n\}$ be a set of n possibly intersecting triangles in 3 space. The *vertical decomposition* of the arrangement $\mathcal{A}(S)$ decomposes each cell of $\mathcal{A}(S)$ into subcells, and is defined as follows (see [1]): from every point on an edge of $\mathcal{A}(S)$—this can be a part of a triangle edge or of the intersection of two triangles—we extend a vertical ray in positive and negative x_3-direction to the first triangle above and the first triangle below this point. This way we create a vertical wall for every edge, which we call a *primary wall*. We obtain a *multi-prismatic decomposition* of $\mathcal{A}(S)$ into subcells, the *multi-prisms*, with a unique polygonal bottom and top face; the vertical projections of both faces are exactly the same. However, the number of vertical walls of a cylinder need not be constant and the cylinder may not be simply connected. We triangulate the bottom face as in the planar case. The added segments are extended upward vertically until they meet the top face. The walls thus erected are the *secondary walls*. Each subcell of the vertical decomposition is now a box with a triangular base and top, connected by vertical walls. (Note that, for navigation purposes, our notion of vertical decomposition is slightly different from other notions of vertical decomposition that construct secondary walls using a planar vertical decomposition of the projections of the top and bottom faces.)

Theorem 4.1 *The vertical decomposition of the arrangement $\mathcal{A}(col \circ H)$ in the work-time space consists of $O(n^2\alpha(n)\log n)$ constant complexity subcells, and can be computed in time $O(n^2\alpha(n)\log^2 n)$.*

Proof: Tagansky [9] proved that the vertical decomposition of the entire arrangement of a set of n triangles in $\mathbb{R}^3$ consists of $O(K + n^2\alpha(n)\log n)$ subcells where K is the complexity of the arrangement. Application of this result to the arrangement of grown obstacle column boundaries $\mathcal{A}(col \circ H)$, which satisfies $K = O(n^2)$, yields the complexity bound.

We can compute the vertical decomposition using an algorithm by De Berg*et al.* [1]. This algorithm runs in time $O(n^2 \log n + V \log n)$, where V is the combinatorial complexity of the vertical decomposition. As $V = O(n^2\alpha(n)\log n)$, the bound follows. $\square$

To faciliate navigation, we want each subcell to have a constant number of neighbors. The common boundary of a subcell κ and one of its neighbors can be a secondary wall, a primary wall, or a 2-face of the arrangement $\mathcal{A}(col \circ H)$. It is easy to see that the number of neighbors sharing a primary or a secondary wall with κ is bounded by a constant. Let us now consider the maximum number of neighbors, sharing a part of a triangle of $\mathcal{A}(col \circ H)$ with κ. Unfortunately the arrangements of walls ending on the top and bottom side of the triangle can be very different, and can in general be as complex as the complexity of the full decomposition which is only upper-bounded by $O(n^2\alpha(n)\log n)$. Simply connecting the subcells at the top of the triangle to the subcells at the bottom of the triangle could result in a number of neighbors that is hard to bound by anything better than $O(n^2\alpha(n)\log n)$ for each subcell κ. However, as we will show, we can connect the subcells at the top and bottom of a face by a symbolic, infinitely thin tetrahedralization. This tetrahedralization will increase the combinatorial complexity of the vertical decomposition by a factor of at most $O(\log n)$, but assures that the number of neighbors per subcell is bounded by a constant. Since this method is quite complicated, we dedicate the following subsection to it. This will lead to the following result:

Theorem 4.2 *There exists a cylindrical decomposition of the base space B for the dynamic low obstacle density motion planning problem consisting of $O(n^2\alpha(n)\log^2 n)$ constant complexity subcells and a constant number of neighbors per subcell. This decomposition can be computed in $O(n^2\alpha(n)\log^2 n)$ time.*

4.2 Tetrahedralizing between Polygons

To reduce the number of neighbors of the subcells we will extend the vertical decomposition with a symbolic connecting structure, that increases the total combinatorial complexity of the vertical decomposition by a factor of $O(\log n)$. As a result, the number of neighbors per subcell of the cell decomposition with the connecting structure will be bounded by a constant. For each face of the arrangement $\mathcal{A}(col \circ H)$, this structure connects the subcells at the top side with the subcells at the bottom side. The structure we use is a symbolic, infinitely thin tetrahedralization. To simplify the discussion, we assume that the face for which we construct the connecting structure is horizontal. (This is not a constraint, but just a matter of definition.) Throughout this section the vertical direction is parallel to the normal of the face.

Both the top and the bottom side of the face contain a triangulated 2-dimensional arrangement, say $\mathcal{T}_t$ and $\mathcal{T}_b$, created by the intersecting faces and the walls that end on it. Such triangulations with extra vertices in their interior are referred to as Steiner triangulations; the extra vertices are called Steiner points. The arrangements $\mathcal{T}_t$ and $\mathcal{T}_b$ are normally different; they do not share Steiner points. We separate the top and bottom of every face in the arrangement. Imagine that the top of the face is at height 1.0 and the bottom at height 0.0. We tetrahedralize the space between the top and bottom arrangement, by adding a number of Steiner points between the top and bottom face. (Remember that this is only done in a symbolic way. In reality, the top and bottom face lie in the same plane. The vertical distance is only used to define the adjacencies of the added (flat) subcells.)

We distinguish between the convex and the non-convex faces. Note that non-convex faces indeed exist, since a column can cut out a part of another column. Theorem 3.4 gives us that the cut out parts are never strictly included in the open interior of a 2-face of a column. We first show how to tetrahedralize the space between two different Steiner triangulations $\mathcal{T}_t$ and $\mathcal{T}_b$ of the same convex simple polygon P.

Our tetrahedralization has two layers joined at height 0.5 by a Steiner triangulation of P. This triangulation $\mathcal{T}_m$ has one Steiner point p: P is triangulated using a star of edges from p to all vertices of P. Both $\mathcal{T}_m$ and $\mathcal{T}_b$ are different Steiner triangulations of the same polygon P, therefore the vertical projections of the boundaries of $\mathcal{T}_b$ and $\mathcal{T}_m$ are equivalent. We tetrahedralize between $\mathcal{T}_m$ and $\mathcal{T}_b$ by adding a face from every edge of $\mathcal{T}_b$ to p. The result is a tetrahedralized pyramid where each tetrahedron corresponds to a triangle of $\mathcal{T}_b$.

To triangulate the complement of this pyramid in the layer between $\mathcal{T}_b$ and $\mathcal{T}_m$, we connect the boundaries of $\mathcal{T}_b$ and $\mathcal{T}_m$ by vertical faces between the boundary edges. For every face f_i introduced by connecting the boundaries, we add a Steiner point q_i in the middle of f_i. We connect q_i to all vertices on f_i and connect each resulting triangle to p (see Figure 3). These triangles complete the tetrahedralization of the space between $\mathcal{T}_b$ and $\mathcal{T}_m$. The tetrahedralization between $\mathcal{T}_t$ and $\mathcal{T}_m$ is constructed in the same way. It is easy to see that the number of tetrahedra created is linear in the complexity of the triangulations $\mathcal{T}_b$ and $\mathcal{T}_t$.

Unfortunately, faces need not be convex. So we must also show how to tetrahedralize the space between two different Steiner triangulations of the same non-convex simple polygon P. (As indicated above we know that the polygon has no holes. This is crucial here.) We again add a Steiner triangulation $\mathcal{T}_m$ of P between $\mathcal{T}_t$ and $\mathcal{T}_b$. In the non-convex case we have to use a more sophisticated Steiner triangulation. For this we use a triangulation by Hershberger and Suri [4] that was originally designed for ray shooting in simple polygons. This triangulation t_m has three important properties. Let k be the number of edges of P:

1. It introduces $O(k)$ Steiner points with each Steiner point directly connected to the boundary of P by at least one triangulation edge;
2. Every line segment that lies inside P intersects at most $O(\log k)$ triangles of $\mathcal{T}_m$;
3. The triangulation can be computed in $O(k)$ time.

We can derive the following lemma from the properties of $\mathcal{T}_m$.

Lemma 4.3 *Let P be a polygon with k vertices and without holes, Let $\mathcal{T}_m$ be the triangulation of P as described in [4]. Let Δ be a triangle inside P, and let $\mathcal{A}_\Delta$ be the*

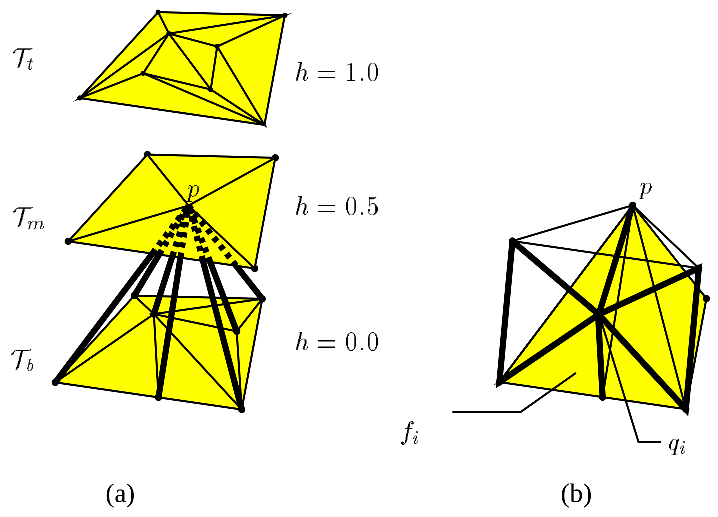

Figure 3: (a) The construction of the pyramid of tetrahedra between p and the triangles of $\mathcal{T}_b$. (The added edges are bold.) (b) The tetrahedra created in the second step between boundary face f_i and the pyramid.

arrangement of t_m inside triangle Δ. Then $\mathcal{A}_\Delta$ has only $O(\log k)$ constant complexity faces.

Proof: Let v be a vertex of $\mathcal{T}_m$ inside Δ. Because, by property (1), every vertex of the triangulation $\mathcal{T}_m$ is connected to the boundary of P by at least one edge, one outgoing edge from v must intersect the boundary of Δ. By property (2), the boundary of Δ intersects at most $O(\log k)$ edges, and, hence, there are also only $O(\log k)$ vertices inside Δ. Since each face of $\mathcal{A}_\Delta$ is the intersection of a triangle from $\mathcal{T}_m$ and the triangle Δ, it has constant complexity. It follows that the whole arrangement $\mathcal{A}_\Delta$ has complexity $O(\log k)$. □

This immediately results in the following corollary.

Corollary 4.4 *Let P be a polygon without holes. Let $\mathcal{T}_m$ be the triangulation described in [4]. Let $\mathcal{T}_a$ be another triangulation of P with complexity m. The arrangement we obtain by overlaying $\mathcal{T}_a$ and $\mathcal{T}_m$ has $O(m \log m)$ constant complexity faces.*

Proof: Each triangle of $\mathcal{T}_a$ is divided into $O(\log k) \leq O(\log m)$ constant complexity faces, by Lemma 4.3. The resulting arrangement therefore has $O(m \log m)$ faces of $O(1)$ complexity. □

Let $\mathcal{T}_{b \times m}$ be the triangulation of P we obtain by overlaying $\mathcal{T}_b$ and $\mathcal{T}_m$ and triangulating the resulting faces. Let m_b denote the complexity of $\mathcal{T}_b$. Corollary 4.4 shows that the complexity of $\mathcal{T}_{b \times m}$ is $O(m_b \log m_b)$. It is easily computed in $O(m_b \log m_b)$ time. To connect $\mathcal{T}_b$ to $\mathcal{T}_m$ (at height 0.5) we place $\mathcal{T}_{b \times m}$ between $\mathcal{T}_b$ and $\mathcal{T}_m$ at height 0.25. First we tetrahedralize the layer between $\mathcal{T}_b$ and $\mathcal{T}_{b \times m}$. We start by adding vertical faces from every edge of $\mathcal{T}_b$ to its corresponding edge in $\mathcal{T}_{b \times m}$. This results in m_b prisms that have the triangles of $\mathcal{T}_b$ as their top and bottom faces. The top faces still contain a number of other edges, that are part of $\mathcal{T}_m$ We tetrahedralize each prism by adding a vertex in the center and connecting it to top, bottom and sides, in the way described for the convex case. (Note that Steiner points might exist on the edges of triangles of $\mathcal{T}_{b\times m}$. However, the triangles on the other side of these edges shares these Steiner points, because they are the result of the intersection of two fully connected arrangements. Therefore each tetrahedralized prism perfectly fits its neighboring prisms.) The number of tetrahedra in this layer is $O(m_b \log m_b)$. We similarly tetrahedralize the layer between $\mathcal{T}_{b\times m}$ and $\mathcal{T}_m$. So the total space between $\mathcal{T}_b$ and $\mathcal{T}_m$ can be filled with $O(m_b \log m_b)$ tetrahedra. In the same way we can fill the area between $\mathcal{T}_t$ and $\mathcal{T}_m$ using the triangulation $\mathcal{T}_{t\times m}$. This tetrahedralization will have $O(m_t \log m_t)$ tetrahedra, where m_t is the complexity of $\mathcal{T}_t$.

Summarizing, we can symbolically create a tetrahedralization between the top and the bottom side of the faces of the arrangement. As the original arrangement has combinatorial complexity $O(n^2\alpha(n)\log n)$ (see Theorem 4.1), the extended arrangement has complexity $O(n^2\alpha(n)\log^2 n)$. It can be computed in $O(n^2\alpha(n)\log^2 n)$ time. The subcells in this arrangement each have constant complexity and a constant number of neighbors.

4.3 The Decomposition of the Free Space

We constructed a decomposition of the work-time space with the following properties:

- The number of subcells is $O(n^2\alpha(n)\log^2 n)$.
- Each subcell has constant combinatorial complexity.
- Each subcell has constant size coverage, i.e. each subcell is intersected by a constant number of columns of grown obstacles.
- Each subcell has a constant number of neighboring subcells.

Since the boundaries of the colums form a subset of the subcell boundaries, it is easy to compute the coverage of each subcell by simply traversing the subdivision of the work-time space. This can, for example, be accomplished by a breadth first search. Now, we can use the same approach as in [11] to compute the complete cell decomposition of the free space (Theorem 2.2). This result is a graph *CG*. Each node of *CG* corresponds to a constant complexity subcell in the free part of the configuration-time space. Each edge corresponds to an adjacency between two such subcells. Here two subcells are called adjacent if and only if they share an f-dimensional face. (This is important because we want to compute free paths, rather than semi-free paths.) The degree of the nodes is bounded by a constant. The complexity of *CG* is the same as the complexity of the base space, so it has $O(n^2\alpha(n)\log^2 n)$ nodes and edges. The computation time of $O(n^2\alpha(n)\log^2 n)$ for the base partition dominates the computation time for *CG*.

5 Finding a Path

In this section we show how to use the cell decomposition to compute a time-monotone path through the free space. Since the path must be time-monotone we cannot do

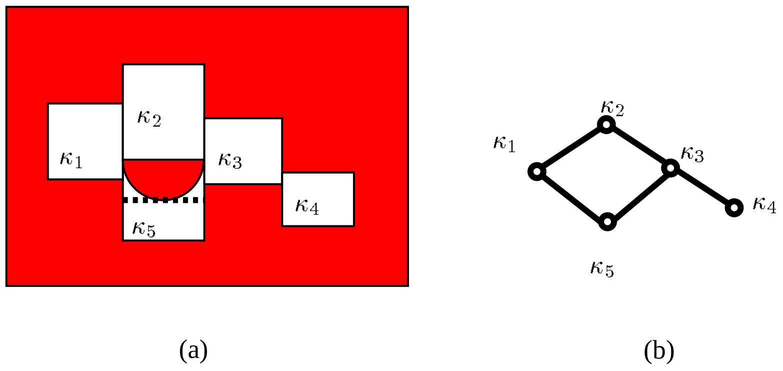

Figure 4: (a) A 2-dimensional space with four connected free subcells. (b) The connectivity graph *CG*. Although there is no time-monotone path from κ_1 to κ_4, this fact is not represented by *CG*.

an arbitrary search through the configuration-time space; we will use a space sweep algorithm in the time direction to keep track of the reachable space, while time passes. We sweep with a hyper-plane $\mathcal{P}$, orthogonal to the time direction, from $t = t_0$ to $t = t_1$. Slightly abusing the notation, we will from now on use *CG* to denote both the connectivity graph and the cell decomposition it represents.

5.1 Preprocessing the Cell Decomposition

To compute the parts of the configuration-time space reachable by time-monotone paths from the start configuration of the robot, we cannot use the cell decomposition directly. The space is partitioned into not necessarily convex constant complexity subcells. The subcells and their adjacencies are represented by the graph *CG*; each subcell in *CG* has a constant number of neighbors. A problem with *CG* is that the time-monotonicity restrictions are not incorporated in the graph. There can be a path between two configurations according to *CG*, while there exists no time-monotone path between those two configurations. Figure 4 shows a 2-dimensional example in which time increases in the vertical upward direction. Although the graph contains a path between the subcells κ_1 and κ_4 there exists no time-monotone path from (any configuration in) κ_1 to (any configuration) in κ_4. Note also that there exists a time-monotone path from κ_1 to only some of the configurations in κ_3.

Since the subcells in the cell decomposition are not necessarily convex, there can even exist a pair of configurations in the same subcell, that cannot be connected by a time-monotone path. In conclusion, the connectivity graph *CG* does not contain all necessary data to find a time-monotone path. It is possible, however, to decompose the $(f + 1)$-dimensional subcells of *CG* into smaller subcells for which there is a time-monotone path for every pair of configurations in the same subcell. If a hyper-plane $\mathcal{P}$, that is orthogonal to the time direction, intersects a subcell κ in a number of disconnected regions, then there might be configurations in κ that cannot be connected with a time-monotone path. We therefore decompose each such subcell κ into a constant number of smaller subcells, such that any cross-section of the hyper-plane $\mathcal{P}$ with a subcell consists of one connected region. If for some t, $\mathcal{P}$ is tangent to a feature of κ, then $\mathcal{P}$ decomposes κ into a constant number of constant complexity subcells. Since κ is of constant complexity, there are $O(1)$ of these tangencies. We decompose κ into $O(1)$ subcells generated by all possible tangencies of $\mathcal{P}$ with features of κ. To adapt *CG*, we replace the node of κ by nodes for the subcells of κ with the appropriate adjacencies. This extension does not increase the asymptotic combinatorial complexity of *CG*. Also the number of neighbors per subcell remains bounded by a (larger) constant. It is easy to see that any pair of configurations in each resulting subcell can be connected by a time-monotone path. In Figure 4 for example, κ_5 is decomposed into three subcells by the dashed line.

5.2 Computing a Path

We take the refined connectivity graph *CG* and note that each subcell κ has the following property: if $(Z, \tau) \in \kappa$ is reachable by a time-monotone path, then all $\{(Z', \tau') \in \kappa | \tau' \geq \tau\}$ are reachable by a time-monotone path. In addition, all adjacent subcells κ' for which the intersection $\kappa \cap \kappa' \cap (t \geq \tau)$ is an f-dimensional face, are (at least partially) reachable through κ.

Our objective is to label each subcell κ with the earliest time τ at which it can be reached from (Z_0, t_0) and with a link to the subcell from which it can be reached at time τ. If the subcell $\kappa_1 \ni (Z_1, t_1)$ receives a label $\tau_1 < t_1$ then (Z_1, t_1) is reachable from (Z_0, t_0) by a time-monotone path. The sequence of subcells containing this path can be found by tracing back the links from each subcell, starting from κ_1.

To obtain the labelling outlined above, we perform a sweep-like search of *CG* starting from $\kappa_0 \ni (Z_0, t_0)$. An event occurs if a subcell κ' is reachable from another subcell κ at some time τ. We keep the events (κ, κ', τ) of the sweep in a priority queue $\mathcal{Q}$—the event with the smallest τ is dequeued prior to every other event.

An event (κ, κ', τ) is handled quite straightforward. Firstly, we add a link from κ' to κ and label κ' with τ. Secondly, we consider each neighbor κ'' of κ' in *CG* for which $\kappa' \cap \kappa'' \cap \{t \geq \tau\}$ is an f-dimensional face. Let $\tau' = \min\{t | \kappa' \cap \kappa'' \cap \{t \geq \tau\} \neq \emptyset\}$. If κ'' is not yet stored in $\mathcal{Q}$ we add the event $(\kappa', \kappa'', \tau')$ to $\mathcal{Q}$. Otherwise, if the time of the event stored for κ'' is later than τ' we replace the old event by the new one.

If the event queue $\mathcal{Q}$ is empty, we are done. We check if the subcell $\kappa_1 \ni (Z_1, t_1)$ received a label. If so, we start in κ_1 and trace the links back to the starting subcell $\kappa_0 \ni (Z_0, t_0)$ to find the (reversed) sequence of subcells that contains a time-monotone free path from (Z_0, t_0) to (Z_1, t_1). If the subcell κ_1 did not receive a label, then (Z_1, t_1) is not reachable by a time-monotone path, and we report failure.

It remains to transform the sequence of subcells into a path. We want this path to lie in the free space (not in its boundary) and we want it to be strictly time-monotone (otherwise the robot would need infinite speed to traverse parts of the path). We call the sequence of connected subcells a *channel*. Because adjacencent subcells have an f-dimensional intersection, the interior of the channel will be connected. Also the interior of the channel lies completely in the free space. Clearly, this interior must contain a strictly time-monotone path. (It contains a time-monotone path by construction, and because the interior is an open space this path can be made strictly monotone.)

In the following, we describe how to first create a semi-free and time-monotone path, and sebsequently transform this path into a free and strictly time-monotone path.

Let κ and κ' be two adjacent subcells in the channel. We create a vertex of the path that lies on the boundary of κ and κ' at the time τ stored with κ'. (τ was the earliest moment at which we could reach κ'.) Inside each subcell we connect the two vertices created with a time-monotone path. Because the subcell has constant complexity this can be done in constant time. The full path is time-monotone but not necessarily strictly time-monotone. Also, it is only semi-free. We have to transform the computed path into a path that is strictly time-monotone and lies completely in the free space. Firstly, we transform the channel into a channel which only consists of subcells with non-zero volume, and contains the original path. This is easily done by tracing the semi-free path and while doing this replacing flat tetraheda by subcells of the original decomposition which contain the path as well. We discard the symbolic coordinates on the fly. Secondly, we transform it into a strictly time-monotone free path by piecewise slightly slanting it in the time direction.

Summarizing, in order to solve a dynamic low obstacle density motion planning problem, we perform the following steps:

```
1. COMPUTE vertical decomposition of $\mathcal{A}(col \circ H)$.
2. for all faces $f$ of $\mathcal{A}(col \circ H)$ do
        COMPUTE flat tetrahedralization between the top and the bottom side of $f$.
3. TRANSFORM $\mathcal{A}(col \circ H)$ into a decomposition of the configuration-time space.
4. COMPUTE a strictly time-monotone free path.
```

It remains to analyse the complexity of the algorithm. Let $|CG|$ denote the size of the graph. Every subcell of the decomposition has a constant number of neighbors and, thus, every subcell creates a constant number of events. This sums up to $O(|CG|)$ events. So the size of $\mathcal{Q}$ is $O(|CG|)$ and enqueueing and dequeueing an event takes $O(\log|CG|)$ time. It is easy to see that handling an event, therefore, takes $O(\log|CG|)$ time. The time for the sweep is therefore upper bounded by $O(|CG|\log|CG|)$. Since $|CG| = O(n^2\alpha(n)\log^2 n)$ by Theorem 4.2, the computation of a time-monotone path takes $O(n^2\alpha(n)\log^3 n)$ time. The following theorem summarizes the result.

Theorem 5.1 *The low obstacle density motion planning problem for a robot among a set of n obstacles that move with constant speed along polylines, can be solved in $O(n^2\alpha(n)\log^3 n)$ time.*

6 Conclusion

In this paper we addressed a dynamic extension of the robot motion planning problem. We developed an approach for the exact motion planning problem for a single robot in a low obstacle density environment with multiple moving objects whose motion is represented by a constant complexity polygonal line. We proved that the complexity of the free space of this motion planning problem is $\Theta(n^2)$. Our algorithm takes $O(n^2\alpha(n)\log^3 n)$ time to compute a free, time-monotone path, for the robot. We are able to construct low obstacle density workspaces, with moving obstacles, for which the path of the robots is of combinatorial complexity $\Omega(n^2)$, so our result is close to optimal. It remains to be seen whether such a bound exists for a robot with bounded velocity modulus, or bounded acceleration.

It is an interesting question whether the results can be extended to a 3-dimensional workspace. We can again define columns in the (now 4-dimensional) work-time space. Theorem 3.3 can easily be extended, leading again to a bound of $O(n^2)$ on the complexity of the free space, like in the two-dimensional case. Also the lowerbound of $\Omega(n^2)$ easily carries over. The problem is to compute a constant-complexity partition of the work-time space that has constant coverage. It is easy to obtain an $O(n^4)$ partition by extending the faces of the columns to hyper-planes, building the complete arrangement of these hyper-planes, and subdividing the resulting cells into simplices. This leads to a close to $O(n^4)$ algorithm for motion planning in dynamic 3-dimensional low obstacle density environments. It is at the moment unclear how to improve this result.

References

[1] M. de Berg, L. Guibas, and D. Halperin. Vertical decomposition for triangles in 3-space. *Discrete & Computational Geometry*, 15:35–61, 1996.

[2] M. de Berg, M. Katz, A.F. van der Stappen, and J. Vleugels. Realistic input models for geometric algorithms. In *Proc. of the 13th ACM Symp. on Computational Geometry* pages 294–303, 1997.

[3] J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proc. 28th IEEE Symp. on Foundations of Computer Science*, pages 49–60, Los Angeles, 1987.

[4] J. Hershberger and S. Suri. A pedestrian approach to ray shooting: Shoot a ray, take a walk. *J. Algorithms*, 18:403–431, 1995.

[5] J. Reif. Complexity of the generalized mover's problem. In *Planning, Geometry and Complexity of Robot Motion*, J.T. Schwartz, M. Sharir, J. Hopcroft (Eds.), pages 267–281, Ablex Pub., Norwood, NJ, 1987.

[6] J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *J. ACM* **41**, 4, pages 764-790, 1994.

[7] J. Schwartz and M. Sharir. On the piano movers' problem: II. general techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4:289–351, 1983.

[8] K. Sutner and W. Maass. Motion planning among time dependent obstacles. *Acta Informatica*, 26:93–133, 1988.

[9] B. Tagansky. A new technique for analyzing substructures in arrangements. In *Proc. 11th Annual ACM Symp. on Computational Geometry*, 1995.

[10] A.F. van der Stappen. *Motion planning amidst fat obstacles*. PhD thesis, Dept. of Computer Science, Utrecht University, 1994.

[11] A.F. van der Stappen, M.H. Overmars, M. de Berg and J. Vleugels. Motion planning in environments with low obstacle density. Technical report, UU-CS-1997-19, Dept. of Computer Science, Utrecht University, 1997.

[12] J. Vleugels. On fatness and fitness – realistic input models for geometric algorithms. PhD-thesis, Dept. of Computer Science, Utrecht University, 1997.