Decomposition-based Motion Planning: Towards Real-time Planning for Robots with Many Degrees of Freedom

Oliver Brock Lydia E. Kavraki
Department of Computer Science
Rice University, Houston, Texas 77005
email: {oli, kavraki}@cs.rice.edu

Abstract

Research in motion planning has been striving to develop faster and faster planning algorithms in order to be able to address a wider range of applications. In this paper a novel real-time motion planning framework, called decomposition-based motion planning, is proposed. It is particularly well suited for planning problems that arise in service and field robotics. It decomposes the original planning problem into simpler subproblems, whose successive solution results in a large reduction of the overall complexity. A particular implementation of decomposition-based planning is proposed. It is based on an adaptive wavefront expansion algorithm and reactive motion execution. Using this implementation of decomposition-based planning, real-time motion planning performance for an eleven degree-of-freedom mobile manipulator can be achieved. Some fundamental and preliminary analysis of the decomposition-based motion planning approach is provided.

1 Introduction

Recent advances in the area of robot motion planning have resulted in the successful application of these techniques to diverse domains, such as assembly planning, virtual prototyping, drug design, and computer animation. Despite these advances, however, some areas of application have still remained out of reach for automated planning algorithms. Applications requiring robots with many degrees of freedom to operate in highly dynamic and unpredictably changing environments fall into that category. In these domains a change in the environment can invalidate the previously planned path. To operate robustly and safely in dynamic environments the ability to modify the planned motion in real-time is necessary. The planning techniques for high-dimensional configuration spaces described in the literature, however, do not generate plans in real-time. In this paper a new planning paradigm is presented addressing these issues by decomposing the planning problem and applying appropriate planning algorithms to the respective subproblems. This results in a real-time planning algorithm in high-dimensional configuration spaces. The planning paradigm is well suited for planning problems of average difficulty, in which a certain amount of clearance to obstacles along a solution path can be assumed. Such planning problems occur frequently in the areas of field and service robotics.

Path planning techniques generally proceed in two phases: in the first phase a representation of the connectivity of the free space is computed; this representation is then used in the second phase to determine a solution to the given path planning problem. Table 1 presents an overview of how various planning methods address these two phases. It is noteworthy that all approaches reduce the second phase to a computationally simple problem: either gradient descent or graph search. This implies that most of the complexity of the path planning problem is addressed in the first phase. Indeed, the potential field approach [15] uses a rather simple and incomplete approach to representing the connectivity and therefore extends well to higher dimensions. The other methods – navigation functions, cell decomposition, and roadmaps – are based on a more sophisticated description of the free space; this is reflected in a computational complexity exponential in the dimensions of the search space [9, 20]. As a consequence, the real-time results achieved with these techniques in low-dimensional configurations spaces [2, 6, 10, 24] do not extend to higher-dimensional planning problems.

Planning Method	**Connectivity Computation**	**Path Generation**
Potential Field [15]	Combine attractive and repulsive potential functions	Gradient descent
Navigation Function [2, 17]	Compute local-minima free potential function	Gradient descent
Cell Decomposition [20]	Determine connected cells representing the free space	Graph search
Roadmap [14, 20]	Determine connected set of low-dimensional curves representing the free space	Graph search

Table 1: Overview of path planning techniques

Much of the recent progress in addressing the problem of real-time performance for path planning in high-dimensional spaces can be attributed to the introduction of probabilistic roadmap techniques [14]. These techniques capture the free space connectivity by randomly sampling the configuration space to find configurations of the robot which are not in collision with the environment. These configurations can be connected to form a probabilistic roadmap. Such a roadmap captures the connectivity of the free space *implicitly*, significantly reducing the computational complexity of its computation compared to explicit methods [20]. By representing the free space implicitly, however, the notion of completeness is lost and can only be replaced with probabilistic completeness [14]. The gain in efficiency observed with probabilistic planners is hence due to a conscious tradeoff between completeness and efficiency.

In what follows we will survey recent accomplishments towards achieving near real-time performance using probabilistic methods. These accomplishments can be divided into three categories:

1. Methods that attempt to reduce the amount of configuration free space that has to be explored, we refer to those as demand-driven;

2. Algorithms that address the narrow passage problem using specialized sampling techniques;

3. Precomputation of possible paths with real-time elimination of invalid ones.

We will elaborate on all of these briefly and then contrast with our approach.

To further reduce the complexity of free space computation in the probabilistic planning paradigm, demand-driven approaches have been developed [3, 19, 22, 29]. Rather than computing the free space connectivity for the entire configuration space before answering a planning query, these methods use heuristics to compute the connectivity in regions of interest, given a specific query. Once the query can be answered, the computation of the free space representation can be aborted.

Randomized techniques have difficulties in finding valid paths in narrow areas of the configuration space [14], because the probability of sampling configurations in free space is low in those areas. Various extensions to probabilistic path planners exploit information derived from workspace geometry to select samples in configuration space to address this problem. Using these extensions, sampling can be performed more efficiently in areas of high or low complexity. In areas of high complexity knowledge about the workspace obstacles can be used to guide the sampling process [1, 12, 13, 25]. The computation of such knowledge, however, tends to be costly. In areas of low complexity the amount of computation can be reduced by discarding redundant samples [21, 28]. Latter approaches require a mapping of workspace information into the configuration space. In high-dimensional configuration spaces such a mapping is either computationally costly or overly conservative [4].

To address planning in dynamic environments an algorithm has been developed that computes a roadmap of a large number of possible paths in configuration space, ignoring all obstacles. The method is influenced by the concepts underlying probabilistic roadmaps. To generate a path, nodes and edges in the roadmap are deleted if they are in collision with the environment [23]. This corresponds to precomputing all possible trajectories in a space without obstacles and eliminating those that are not feasible, given the current environment. This is the first path planning method to address the problem of real-time path planning.

The planning framework presented in the remainder of this paper takes a fundamentally different approach from the planning methods described above. Decomposition-based planning addresses planning problems of average difficulty with robots of many degrees of freedom. Average difficulty refers to planning problems for which the robot moving along a solution path can be assumed to have a certain clearance to obstacles. Such planning problems occur frequently in field and service robotics. Decomposition-based planning is not well suited for more difficult problems like the alpha puzzle [1], where a solution path leads through very narrow passages in configuration space, or even motion in contact with the environment. Decomposition-based planning decomposes the planning task into subproblems of non-trivial complexity, as opposed to one difficult and one trivial problem, as discussed in Section 1 and laid out in Table 1. Real-time motion planning for robots with many degrees of freedom can be achieved by using the solution of one sub-problem as a seed to solving the overall planning problem. The planning process accomplishes a tight integration of planning and control algorithms, similarly to previous work in high-dimensional [7, 28] and low-dimensional configuration spaces [6].

2 Decomposition-based Motion Planning

This section introduces decomposition-based planning. As mentioned before, decomposition-based planning is a motion planning framework addressing motion planning problems of average complexity, like those encountered in field and service robotics. In the application areas certain assumptions about the amount of the free space around the robot at a given configuration can be made. Decomposition-based planning is best suited for planning problems in which a minimum clearance to obstacles can be assumed.

Rather than presenting a particular implementation of decomposition-based planning, this section introduces a framework. Each part of the framework can be performed with a wide range of methods and algorithms that depend on the particular planning problem at hand. In what follows we discuss the underlying principles of the framework. In Section 3 the framework is then applied to a concrete planning problem and the particular implementation choices are laid out in detail.

2.1 Motivation

The discussion of recent path planning methods aiming at real-time performance in Section 1 reveals the focus on finding more efficient ways of computing free space connectivity. Connectivity is represented in such a manner that determining a path becomes a simple graph search problem. This implies that connectivity is computed and represented in the same space as the motion of the robot – in the configuration space.

The underlying assumption of decomposition-based motion planning is that connectivity information about the free space can be computed and represented more easily in a low-dimensional space, while the motion of the robot must be generated in a high-dimensional space, namely the configuration space associated with the robot. This naturally leads to a decomposition of the overall planning task into a global and a local problem. The global problem is to capture as much of the connectivity of the free space as is needed to solve the original planning problem; the local problem is to find a motion for the robot, given that connectivity information. The solution to the local problem must be represented in the high-dimensional configuration space of the robot, as it must represent a valid motion.

As was noted in Section 1, probabilistic methods have successfully traded completeness for efficiency, by replacing the notion of completeness with probabilistic completeness. As a result, the narrow passage problem arose [11, 13], referring to the difficulty of probabilistic methods to find solution paths through narrow passages of the configuration space. Decomposition-based planning performs a tradeoff similar to probabilistic methods, giving up completeness for efficiency by decomposing the planning problem into two subproblems. As a result, this method exhibits a problem similar to the narrow passage problem, discussed in more detail in Sections 2.2 and 2.3.

Given a planning problem P of dimension d, decomposition-based planning decomposes P into planning problems P_1 and P_2 of dimensionality

$0 < d_1 < d_2 = d$, such that the solution to P_1 can be used as an aid for efficiently finding a solution to P_2. The goal of the decomposition of P is to find subproblems for which a solution can be obtained very efficiently. Note that P_1 needs to capture as much of the relevant global connectivity of the free space as possible, whereas the solution to P_2 has to represent a valid solution to the original planning problem P.

Underlying any decomposition in this planning paradigm is the realization that the dimensionality of the solution space is larger than the dimensionality of the problem space. The planning problem is defined by constraints in the workspace $\mathcal{W}$ of dimension $d_1 \leq 3$; the solution on the other hand is represented in the configuration space $\mathcal{C}$ of much higher dimension $d_2 > 3$. The additional dimensions arise due to the kinematic constraints of the robot. The high-dimensional solution in $\mathcal{C}$ is connected to the low-dimensional problem space $\mathcal{W}$ via the workspace volume V swept by the robot along its trajectory defined in $\mathcal{C}$. In other words, a path or trajectory in a high-dimensional space can be represented as volume in the low-dimensional workspace. Decomposition-based planning uses this connection between the two spaces to divide the planning task, as will be explained in detail below.

2.2 Framework

Consider a planning problem P for a robot R in a configuration space $\mathcal{C}$ of dimension d, with an initial configuration $\mathbf{q}_{init}$ and a final configuration $\mathbf{q}_{goal}$. Assume there exists a path τ from $\mathbf{q}_{init}$ to $\mathbf{q}_{goal}$ entirely in the free space $\mathcal{F} \subseteq \mathcal{C}$. Then let V_τ denote the workspace volume swept by the robot R along the path τ. For now we consider the workspace $\mathcal{W}$ to be the Euclidean space $\mathbb{R}^3$. Furthermore, let $H(\tau)$ denote the set of all paths homotopic to the path τ. The workspace volume $V_{H(\tau)}$ is then defined as

$$V_{H(\tau)} = \bigcup_{\sigma \in H(\tau)} V_\sigma,$$

representing the combined workspace volume swept along all paths homotopic to τ.

Let us assume that there are n homotopically distinct solution paths $\tau_i, 1 \leq i \leq n$ to the planning problem P. The set of all solution paths $S(P)$

to P is then given by

$$S(P) = \bigcup_{1 \leq i \leq n} H(\tau_i).$$

For any given solution path τ the relation $V_\tau \subseteq V_{H(\tau)} \subseteq V_{S(P)}$ must hold.

We define $V_\tau^\delta = V_\tau \oplus b(\delta)$, where $\oplus$ denotes the Minkowski sum and $b(\delta)$ denotes a ball of radius δ, to represent the volume swept by the robot along the path τ grown by δ. The planning problem P is said to be δ-hard if there exists a path $\tau \in H(\tau_i)$ such that $V_\tau^\delta \subseteq V_{H(\tau_i)} \subseteq S(P)$. This means that at every point along the path τ the robot has at minimum a clearance of δ from the closest obstacle. The decomposition-based planning approach presented here addresses planning problems that are δ-hard.

We want to decompose the planning problem P into two subproblems, P_1 and P_2. The planning problem P_1 can be defined as determining a workspace volume T, called *tunnel*, such that $V_\tau \subseteq T$ for at least one solution path τ. Since τ and therefore V_τ are not known, however, a simplified criterion has to be used to ensure the tunnel T is computed in a manner that $V_\tau \subseteq T$. Such a criterion is called complete if for every solution path $\tau \in H(\tau_i) \subseteq S(P)$ the relation $V_\tau \subseteq T$ holds. Note that T might also represent paths σ that are not solution paths, $\sigma \notin S(P)$. This means that a connected component in T might actually not be connected in the free configuration space $\mathcal{F} \subset \mathcal{C}$. In other words, we give up the notion of soundness of the criterion with respect to the the original planning task. It is easy to regain that soundness by making conservative assumptions about P_1. Whether such a complete criterion can be computed efficiently depends on the planning problem at hand. Alternatively, an *incomplete* criterion can be used, meaning that there are solution paths τ such that $V_\tau \not\subseteq T$. Such a criterion can be computed much more efficiently, but introduces incompleteness. In choosing an incomplete criterion the tradeoff between completeness and efficiency needs to be considered carefully. In the remainder of this paper we will be concerned with methods that find a solution path $\tau \in H(\tau_i)$ if $V_\tau^\delta \subseteq V_{H(\tau_i)} \subseteq S(P)$, for a given value of δ. These methods are called δ-complete. There are many such methods and the optimal choice depends on the problem at hand. Section 3.1 introduces one such method, addressing the planning problem for mobile manipulators.

Once we have obtained a workspace volume T, we define the second planning problem P_2 to consist of finding a path $\tau \in S(P)$ such that $V_\tau \subseteq T$. Again, various planning methods can be employed to accomplish this task; a particular one is presented in Section 3.2 in the context of motion planning for mobile manipulators.

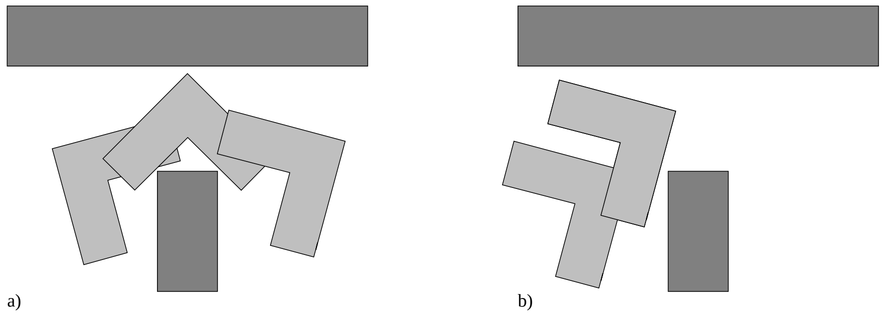

Figure 1: Two different paths for an L-shaped object in the plane occupying the same free space.

In finding a path τ, using the workspace volume T as a guide in the search space, however, a difficulty is encountered. Let τ be the solution path to a planning problem P, and let $H(\tau) = S(P)$, i.e., there exists only one set of homotopically equivalent solution paths. Given a volume T, we are trying to determine a path $\sigma \in H(\tau)$. However, a path σ might exist such that $V_\sigma \subseteq T \subseteq V_{S(P)}$ without σ representing a solution to P: $\sigma \notin H(\tau) = S(P)$. This situation arises due to distinct configuration space paths sweeping an overlapping volume of workspace. An example of such a situation is shown in Figure 1. Part a) of that figure shows how an L-shaped object passes a narrow passage in the workspace along a path τ. The area $V_{H(\tau)}$ consists of the shown free space. Part b) of this figure shows a path σ, which is not a solution path as it fails to pass the narrow passage. However, for the initial part of τ and σ the volume swept by their respective homotopic equivalence classes are overlapping. During the search for a solution path given the workspace volume T, this property makes it impossible to distinguish between τ and σ until the narrow passage is reached. Depending on the particular planning problem at hand the method chosen to solve P_2 needs to address this difficulty.

So far the workspace $\mathcal{W}$ was assumed to be the Euclidean space $\mathbb{R}^3$. It is worth mentioning that the framework can directly be applied to $\mathbb{R}^1, \mathbb{R}^2$, and $\mathbb{R}^3 \times t$, where t denotes the time axis.

In this section the framework for decomposition-based planning was introduced. A planning problem P is divided into subproblems P_1 and P_2. The subproblems are connected to each other via the workspace volume swept by a set of solution paths in the workspace. Specific properties and problems of P_1 and P_2 were discussed. Section 3 uses the proposed framework to solve planning problems in the domain of mobile manipulators.

2.3 Discussion

Decomposition-based planning is motivated by the fact that solution paths for most planning problems encountered in service and field robotics, and even many problems in manufacturing, have a relatively large clearance to obstacles along almost the entire path. This lead to the definition of δ-hardness. The novel planning paradigm presented here attempts to solve such problems in real-time by trading completeness, as understood by the notion of δ-hardness, for efficiency.

The tradeoff of completeness for efficiency is a result of two assumptions made during the decomposition of the original planning problem. Let $\tau \in H(\tau_i)$ represent a solution path to the original planning problem. The approximation of $V_{H(\tau_i)}$ by T relies on the assumption that there exists a path $\tau \in H(\tau_i) \subseteq S(P)$ such that $V_\tau \subseteq T \subseteq V_{S(P)}$. For most practical algorithms the volume of T is going to be a proper subset of $V_{S(P)}$, i.e. $T \subset V_{S(P)}$. This approximation is addressed by the notion of δ-completeness.

The more significant loss of completeness is incurred by the fact that a solution path τ and a non-solution path σ might be indistinguishable according to the volume swept along the paths in their homotopic equivalence classes, $V_{H(\tau)}$ and $V_{H(\sigma)}$. There are classes of robots for which this problem does not arise, like convex, symmetric robots, for example. If the problem does arise, it can either by addressed by specific local planning methods or by assuming a large δ for the δ-hardness of the problem. In either case it needs to be noted that the incompleteness of decomposition-based planning is a conscious choice in the pursuit of real-time performance for planning algorithms in high-dimensional configuration spaces.

Some planning approaches presented in the literature exhibit ideas that are reminiscent of decomposition-based planning. A particular low-dimensional instance of decomposition-based planning was applied to the problem of planning for a robot moving in the plane [10]. The idea of decomposing the planning task into capturing a volume in space and imposing a navigation function onto that space can also be found in an approach to planning feedback motion strategies [28]. Here, the volume of free space is computed in configuration space, resulting in larger computational complexity. Other planning approaches use projection to reduce the complexity of the planning problem; these approaches assume that a solution to P_1 of the decomposition automatically is a solution to P_2 [18, 24]. Finally, the idea of dimensionality reduction of the planning problem can be traced back to the silhouette method [8], where the planning problem is recursively projected into lower dimension. This particular approach, however, differs significantly in the way the subproblems are treated.

In the next section we will demonstrate that decomposition-based planning accomplishes real-time planning in a relatively wide range of realistic problems for robot with many degrees of freedom.

3 A Decomposition-based Planning Algorithm

This section presents a decomposition-based planning algorithm. The goal is the development of a real-time planning algorithm for mobile manipulators with many degrees of freedom. As described in Section 2, the planning problem is decomposed into two subproblems. The first subproblem P_1 of identifying a tunnel T will be addressed by a wavefront expansion algorithm for free space computation. The second subproblem P_2 of determining a solution path in the configuration space will be solved using potential field techniques and a navigation function, resulting from the solution of the first subproblem P_1.

3.1 Solving P_1: Wavefront Expansion

As outlined in Section 2 the subproblem P_1 consists of determining the workspace volume T, called tunnel, such that the volume V_τ swept by the robot along a solution path τ is contained within T, i.e. $V_\tau \subseteq T \subseteq V_{H(\tau)}$. In this particular instantiation of decomposition-based planning, the tunnel T will be determined by a wavefront expansion algorithm described in this section.

Wavefront expansion algorithms proceed by expanding a virtual, omni-directional wave starting from the goal configuration of the robot. In expanding, the constraints imposed by obstacles are observed. During the expansion, every cell is labeled with the time value for when it was reached by the virtual wave. Cells close to the goal will have small time values associated with them, cells further away will have larger values. The labels of the individual cells form a discrete, local minima free potential function.

Wavefront expansion algorithms presented in the literature [2, 20] divide the space into a uniform grid. By choosing a uniform discretization, the granularity required to accurately sample the most difficult region of the space is imposed on the entire space. The adaptive wavefront expansion algorithm presented here operates in the workspace and adjusts the size of the cells to the local geometry. Each cell is a spherical workspace region of free space centered around a point p; this region is called bubble [27] and contains all points $\mathbf{q}$ that are closer to p than the closest obstacle. A bubble is defined as $\mathcal{B}(p) = \{\, q : \|p-q\| < \rho(p)\}$, where $\rho(p)$ computes the minimum distance from p to any obstacle.

The goal is to compute an expanding wavefront consisting of bubbles. The wavefront is maintained in a tree structure with the root being the bubble around the goal configuration. Immediate children of a node represent neighboring cells as bubbles of free space. As the wave propagates from the initial bubble, children are inserted into the tree. The levels of the tree can intuitively be viewed as instances of the wavefront at different points in time. This is not exactly accurate, as the cell size and therefore the speed of expansion will depend upon the geometry of the workspace. The wavefront is starting from a starting point s in the workspace; its computation terminates once a goal location g is reached. This procedure has to be slightly adapted to the shape of the robot to guarantee that the initial and the final configuration of the robot are contained within the volume of free space computed by this procedure.

The algorithm proceeds as follows: We compute the radius r of bubble $\mathcal{B}_s$ centered at the start configuration s for the wavefront expansion. This bubble is inserted into a priority queue, prioritized by the minimum distance between the bubble and the goal location g. With its center p and radius r we store parent $\mathcal{B}_p$ of the bubble, which in this case is the empty set $\emptyset$. If the goal location is designated by g, the priority value according to which the bubble is inserted into the priority queue is given by $\|p - g\| - r$. This represents a best-first planning approach: the bubble nearest to the goal configuration has the highest priority.

The algorithm now iterates until either a termination criterion is met, indicating that a path has been found, or the priority queue is empty. Each iteration begins by removing the bubble $\mathcal{B}$ with the highest priority from the queue and inserting into the tree V as a child of its parent, representing the currently explored free space. The surface of $\mathcal{B}$ is randomly sampled; if the sample is not contained in other bubbles in the previously explored free space, the bubbles centered at those samples are computed. Those bubbles are inserted into the priority queue and the process is repeated. This algorithm is laid out in Figure 2. The parameters of this algorithm, such as the number n of samples can be varied; the algorithm is not very sensitive to those parameters.

Figure 3 shows a sequence of four pictures, each consisting of three images, documenting the incremental computation of the free space representation, using the algorithm described above. The images represent two different views of the same scene. The two leftmost images show in white the tree structure by connecting the centers of a parent and its children by lines. The rightmost images show the bubbles representing the approximated free space.

If Figure 3 a) the initial configuration of the robot can be seen. The robot is a free-flying Mitsubishi PA-10 with eleven degrees of freedom. The goal configuration for the robot is indicated by a single white line and a single sphere. This configuration is the starting configuration s for the wavefront algorithm; the goal configuration g is the current position of the end-effector.

```
procedure wavefront expansion(s, g, n)
    initialize priority queue Q
    initialize tree V
    compute B = (s, r, ∅)
    insert B into Q with priority ||s − g|| − r
    while Q not empty and g not reached
        pop Q into B_p = (p, r, B_parent)
        insert B_p = (p, r, B_parent) into v with parent B_parent
        determine n points p_i on surface of B by random sampling
        discard p_i ⊂ B_i ∈ Q
        compute B_{p_i} = (p_i, r, B)
        insert B_{p_i} = into Q with priority ||p_i − g|| − r
    end while
    if s and g are connected then
        return V
    end if
    return failure
end procedure
```

Figure 2: Pseudo-code for the wavefront expansion algorithm.

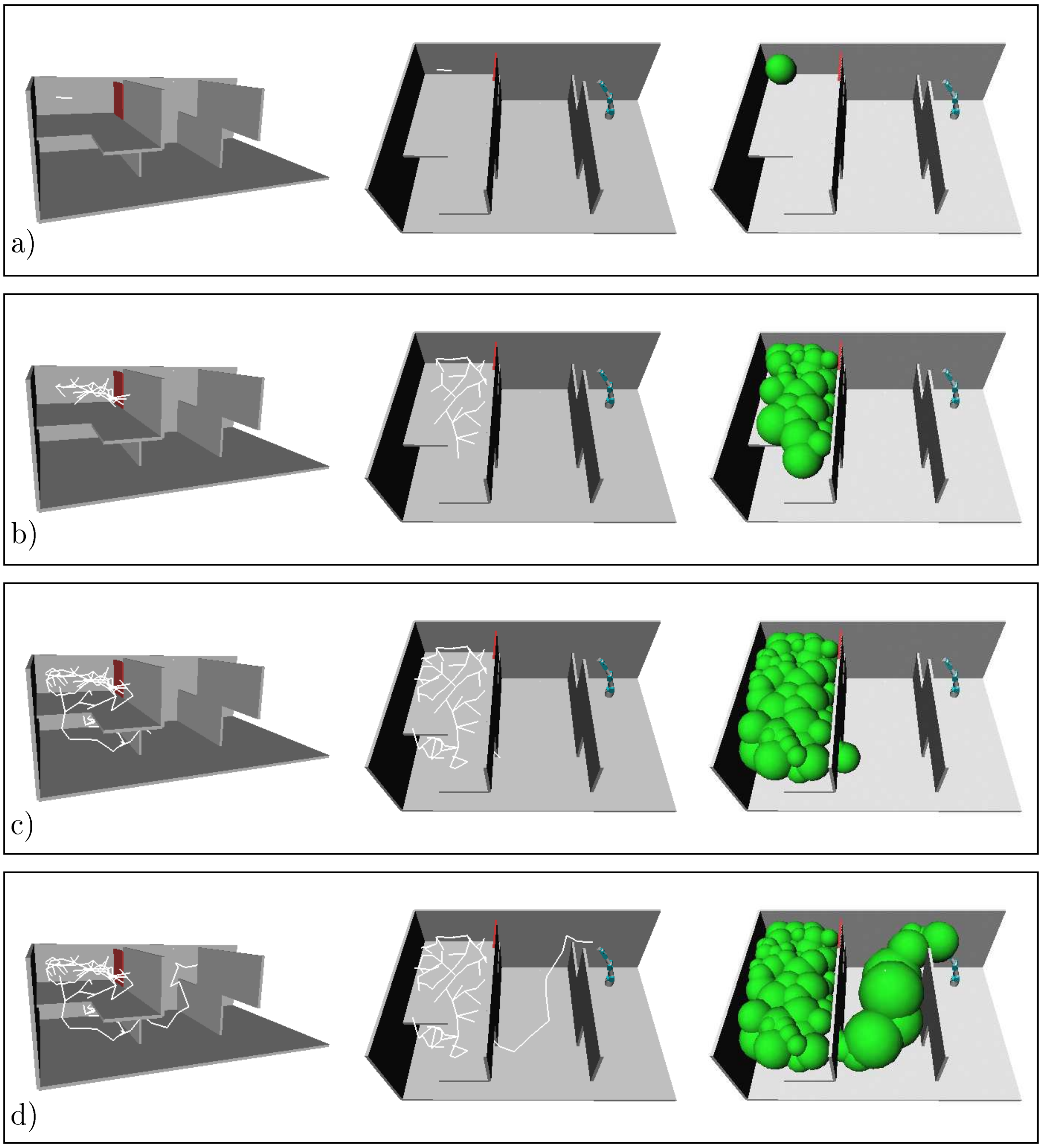

Figure 3: Wave-front expansion to determine a tunnel T.

Figure 3 b) shows the computed free space after the first local minimum has been explored. Due to the adaptive nature of the wavefront expansion, however, this can be done very efficiently. The next image shows the free space representation after the local minimum on the lower floor has been explored. The final image in Figure 3 d) shows the completed free space representation. It is clearly visible how the adaptive nature of the exploration results in large bubbles in those parts of the workspace containing large, continuous regions of free space. The heuristic of expanding the bubble containing the closest point to the current position to the end-effector first results in an efficient, demand-driven exploration.

The computational complexity of the wavefront expansion algorithm described above can empirically be determined to be roughly proportional to the complexity of the environment. This is explained by the adaptive nature of the algorithm. Large areas of free space are rapidly explored by large bubbles. Narrow areas in the workspace require an increasing number of bubbles.

The minimum size of the bubble contributing to T and the amount of overlap between adjacent bubbles needs to be chosen appropriately for a given δ-hard problem. We assume the existence of a solution path τ such that $V_\tau^\delta \subseteq V_{S(P)}$. This means in order to capture V_τ in T we cannot underestimate $V_{S(P)}$ by more than δ. The particular choice of δ and the parameters that influence it depend on the planning problem.

The wavefront expansion algorithm in most cases computes a workspace volume much larger than the desired tunnel T. As significant portions of the free space have to be explored to determine its connectivity, areas not contributing to T are also added to the tree, as described above. The algorithm used to solve planning problem P_2, presented in Section 3.2 will address this issue.

Other algorithms could be used to represent a free space volume in the workspace. One natural choice would be a trapezoidal decomposition, which can be easily obtained from a geometric model of the environment using standard algorithms from computational geometry. The adaptive wavefront algorithm presented above was chosen, however, because it integrates nicely with changing environments perceived via sensors and abstracts from surface complexity of objects in the environment. If sensors are used to derive a model of the environment, its representation is subject to degeneracies, complicating a standard trapezoidal decomposition, or might not even be in the form of topological information, but rather in the form of a point cloud. Furthermore, the adaptive wavefront algorithm allows the demand-driven exploration of the workspace. The framework of decomposition-based planning, however, in not restricted to the use of either algorithm.

3.2 Solving P_2: Artificial Potential Fields

Using the tunnel T computed by solving P_1, we now determine a path τ for the robot. This will be accomplished by imposing a local-minima free potential function on the free space representation determined by the wavefront expansion algorithm. Artificial potential field methods will be applied to determine a valid path in the configuration space $\mathcal{C}$ of the robot.

3.2.1 Navigation Function

Artificial potential fields [15] are an efficient approach to motion generation in dynamic environments. They suffer from local minima, which may prevent the robot from reaching its desired goal configuration. To address this problem, the notion of navigation function was introduced [17]. Navigation functions are local minima-free potential functions. In general, these functions are difficult to compute analytically. By discretizing the space in which these functions are computed, their computation can performed numerically, leading to numerical navigation functions, such as the wavefront expansion algorithm [2, 20].

Section 3.1 introduced an adaptive wavefront expansion algorithm for the incremental computation of a volume in three-dimensional workspace. The history of this wavefront expansion is represented as a tree and can be easily used to impose a local minima free navigation function. Such a navigation function $\mathcal{N}$ consists of a whole family of local minima-free potential functions $\mathcal{N}_1, \ldots, \mathcal{N}_k$, one for each path from a leaf node of the tree to its root. The navigation function for a given path within the tree between a leave node and the root node, given by a sequence of bubbles $\mathcal{B}_l, \ldots, \mathcal{B}_i, \ldots, \mathcal{B}_1$, can be defined in terms of the distance to the goal. The distance from a center of a bubble to the goal is measured along the line segments connecting the centers of consecutive bubbles along the path and is designated by $\rho(\mathcal{B}_i)$. The navigation function $\mathcal{N}_j$ for the sequence of bubbles $\mathcal{B}_l, \ldots, \mathcal{B}_i, \ldots, \mathcal{B}_1$ at point p is then given by:

$$\mathcal{N}_j(p) = \begin{cases} \|p - c_1\| & \text{if } p \in \mathcal{B}_1 \\ \|p - c_i\| + \rho(\mathcal{B}_i) & \text{if } p \in (\mathcal{B}_{i+1} - \mathcal{B}_i) \\ \infty & \text{otherwise} \end{cases},$$

where c_i designates the center of bubble $\mathcal{B}_i$.

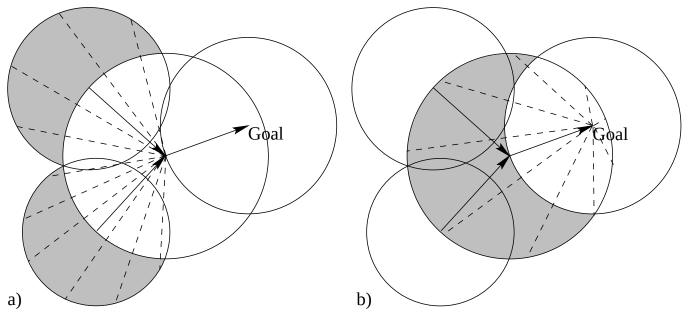

Figure 4: Defining the numerical navigation function.

Figure 4 illustrates the definition of the navigation function, given a tree of bubbles. In part a) and b) of that figure the gray regions indicate the set of points for which the navigation function is shown, the dashed lines indicated the direction of the gradient of the navigation function, and the arrows show the parent relationship between bubbles. A point contained in the gray region of Figure 4 a) will follow the gradient of $\mathcal{N}$ until it enters the shaded area in part b) of that figure. Figure 4 b) shows the potential a point would be exposed to after entering the gray region. This traversal continues until the bubble containing the goal location is reached.

Classical numerical navigation [2, 20] functions were computed starting from the goal. This allowed a navigation function to remain valid for all points in the space as long as the environment did not change. Since we assume the environment to be dynamic, we recompute the navigation function for each motion command. Therefore, the computation of the navigation function presented here could equally well start from the current configuration of the robot. This would allow the robot to move before a valid path to the goal has been computed.

Imposing a navigation function as described above, effectively allows the selection of a tunnel T from the overall free space representation obtained by the wavefront expansion algorithm described in Section 3.1. Let $\mathcal{B}_s$ and $\mathcal{B}_g$ designate the bubbles connecting the starting and goal configuration of the wavefront. For the sake of simplicity we will assume that the robot is entirely contained within those bubbles. Furthermore, let the set $\mathcal{B}_{s,g}$ designate all bubbles along the path in the tree connecting $\mathcal{B}_s$ to $\mathcal{B}_g$. The tunnel T is then defined as

$$T = \bigcup_{\mathcal{B}_i \in \mathcal{B}_{s,g}} \mathcal{B}_i.$$

By following the gradient of the navigation function imposed on this tunnel T as described above, a point robot will be guided through the tunnel T. To extend this concept to an articulated robot arm, we need to resort to the notion of control points [15]. The next section will detail how this can be accomplished in an efficient manner.

3.2.2 Reactive Motion Execution

Reactive motion planning was first introduced in the artificial potential field approach [15]. This approach has been integrated with motion planning to result in the elastic strip framework [5, 7]. In this approach, real-time path modification is achieved by reactively updating a path, previously generated by a planner. The original path is modified by combining task behavior and obstacle avoidance, allowing continuous execution of a task, while reactively avoiding obstacles. Here, we are concerned with the more difficult problem of determining the path or trajectory, rather than just executing a precomputed one.

For a robot to react to obstacles in the environment, proximity information needs to be translated into joint motion. Such proximity information can be easily obtained by distance computation in the workspace. As a result, a virtual force $\mathbf{F}$ can be computed, indicating a direction and a magnitude of force acting on the robot caused by a nearby obstacle. This force $\mathbf{F}$ can then be translated into joint torque $\mathbf{\Gamma}$ using the Jacobian J at configuration $\mathbf{q}$ of the robot: $\mathbf{\Gamma} = J^T(\mathbf{q})\mathbf{F}$. This effectively maps the low-dimensional force vector $\mathbf{F}$ from the workspace into the high-dimensional joint space of the manipulator. Using this mapping reactive obstacle avoidance can be achieved.

During the execution of a task by a robot, it is desirable to link reactive obstacle avoidance with task execution. This is particularly relevant in situations where the task to be accomplished requires fewer degrees of freedom than the robot has. For example, an eleven degree-of-freedom robot positioning an object in space only requires three degrees of freedom to accomplish this task; the remaining degrees of freedom can be used for obstacle avoidance during execution of the task.

The framework for combining task behavior and obstacle avoidance behavior relies on the general structure for redundant robot control. In this structure the torques $\mathbf{\Gamma}$ that are applied to the robot are computed as follows:

$$\mathbf{\Gamma} = J^T(\mathbf{q})\mathbf{F} + \left[I - J^T(\mathbf{q})\overline{J}^T(\mathbf{q})\right]\mathbf{\Gamma}_0 \tag{1}$$

[16], where J is the Jacobian of the manipulator, $\overline{J}$ designates its dynamically consistent pseudo inverse, $\mathbf{F}$ describes the forces defined by the task, and $\mathbf{\Gamma}_0$ denotes the torques to implement obstacle avoidance. Equation 1 provides a decomposition of of the joint torques into those caused by forces at the end effector ($J^T\mathbf{F}$) or operational point and those that only affect internal motion of the robot $\left(\left[I - J^T(\mathbf{q})\overline{J}^T(\mathbf{q})\right]\mathbf{\Gamma}_0\right)$. This decomposition can be exploited to use task-independent degrees of freedom of the robot for obstacle avoidance in the nullspace. Simple obstacle avoidance without the incorporation of task behavior can be achieved by mapping attractive and repulsive forces to joint torques using equation

$$\mathbf{\Gamma} = J^T(\mathbf{q})\mathbf{F}. \tag{2}$$

Here, the forces $\mathbf{F}$ are the combination of forces to accomplish the task $\mathbf{F}_{task}$ and forces $\mathbf{F}_{obst}$ to avoid obstacles: $\mathbf{F} = \mathbf{F}_{task} + \mathbf{F}_{obst}$. Since there is no decoupling, obstacle avoidance behavior can affect task execution behavior.

In Section 3.2.1 a numerical navigation function was imposed on the tunnel T of workspace computed using the algorithms described in Sections 3.1 and 3.2.1. We can define the gradient of this navigation function as the basis for deriving $\mathbf{F}_{task}$, when the task is to reach a certain position in the workspace with the end-effector of the robot:

$$\mathbf{F}_{task} = -\nabla\mathcal{N},$$

where $\mathcal{N}$ is the adaptive numerical navigation imposed on the tunnel T. The gradient of $\mathcal{N}$ defines the task to be executed by the robot. Combining the forces resulting from $\mathcal{N}$ with repulsive forces $\mathbf{F}_{obst}$ derived from proximity information to obstacles allows the real-time computation of a solution to subproblem P_2. To compute the solution to P_2 efficiently, the solution to subproblem P_1 as represented by T after augmentation with the navigation function $\mathcal{N}$ is exploited.

In certain situations following the gradient of $\mathcal{N}$ might not lead to a solution for the planning problem P_2. This problem arises for structural local minima of the robot, such as the one shown in Figure 1, and is a result of the conscious tradeoff of completeness for efficiency. Methods allowing to minimize the impact of structural minima need to be developed.

Note that this particular implementation of decomposition-based planning not only determines a solution path to the given planning problem, but implicitly defines a trajectory in real-time. This is a significant advantage over other planning approaches, where subsequent to the path planning process a time-parameterization has to be imposed onto the resulting solution path.

4 Experimental Results

The real-time motion planning algorithm described above was implemented on a 175MHz SGI O2. It was applied to an eleven degree-of-freedom manipulator, consisting of a free-floating base with four degrees of freedom and a Mitsubishi PA-10 manipulator arm with seven degrees of freedom. The experimental setup can be seen in Figure 5.

Depending on the complexity of the environment and the size of its local minima, the computation of the wavefront expansion algorithm was performed at rates between 3 and 100 Hz. The computational complexity of this procedure is governed by the distance computations necessary to determine the size of the free space bubbles. These distance computations can be performed efficiently, using hierarchical bounding spheres [26] to represent the environment. It is interesting to note that the number of bubbles necessary to cover a rectangular parallelepiped of free space with a given precision is dependent upon the aspect ratio of the volume [4]. The computation of the tunnel T and the numerical navigation function can be performed in parallel with the control loop for reactive motion generation of the robot, resulting in a very tight integration of planning and control. Each time a new solution to P_1 becomes available, the control loop for P_2 uses the new navigation function to determine the motion of the robot.

Figure 5 shows a series of snapshots from a preliminary implementation of the algorithms described above. The motion execution depicted required four planning operations. Figure 5 a) shows the environment, the robot in its initial position, and the initial result of the adaptive wavefront expansion algorithm, shown as a branching tree-like graph in space, with its root at the goal configuration for the end-effector. For these preliminary results the path was not smoothened. A real-time smoothing operation could be performed using the elastic strip framework [5]. In part b) of the figure an obstacle is blocking the original path for the end-effector and a new free space representation and navigation function are computed, as can be seen in c). Figures 5 d) and f) show the result of subsequent real-time computations of the navigation function, following invalidation by an unforeseen obstacle. Note that repulsive forces originating from obstacles in the environment cause the robot to avoid collisions in a reactive manner, as can be seen in Figures 5 d) and e), where the robot passes a narrow region of free space. All degrees of freedom of the robot are used to avoid the obstacles.

Using the decomposition-based planning approach presented in this paper, real-time planning was implemented for an eleven degree-of-freedom manipulator. Except for the mapping given in equation 2 no part of the computation is dependent upon the degrees of freedom of the robot, but rather on its geometric properties and the properties of the workspace. Therefore these results are expected to scale to very high number of degrees of freedom.

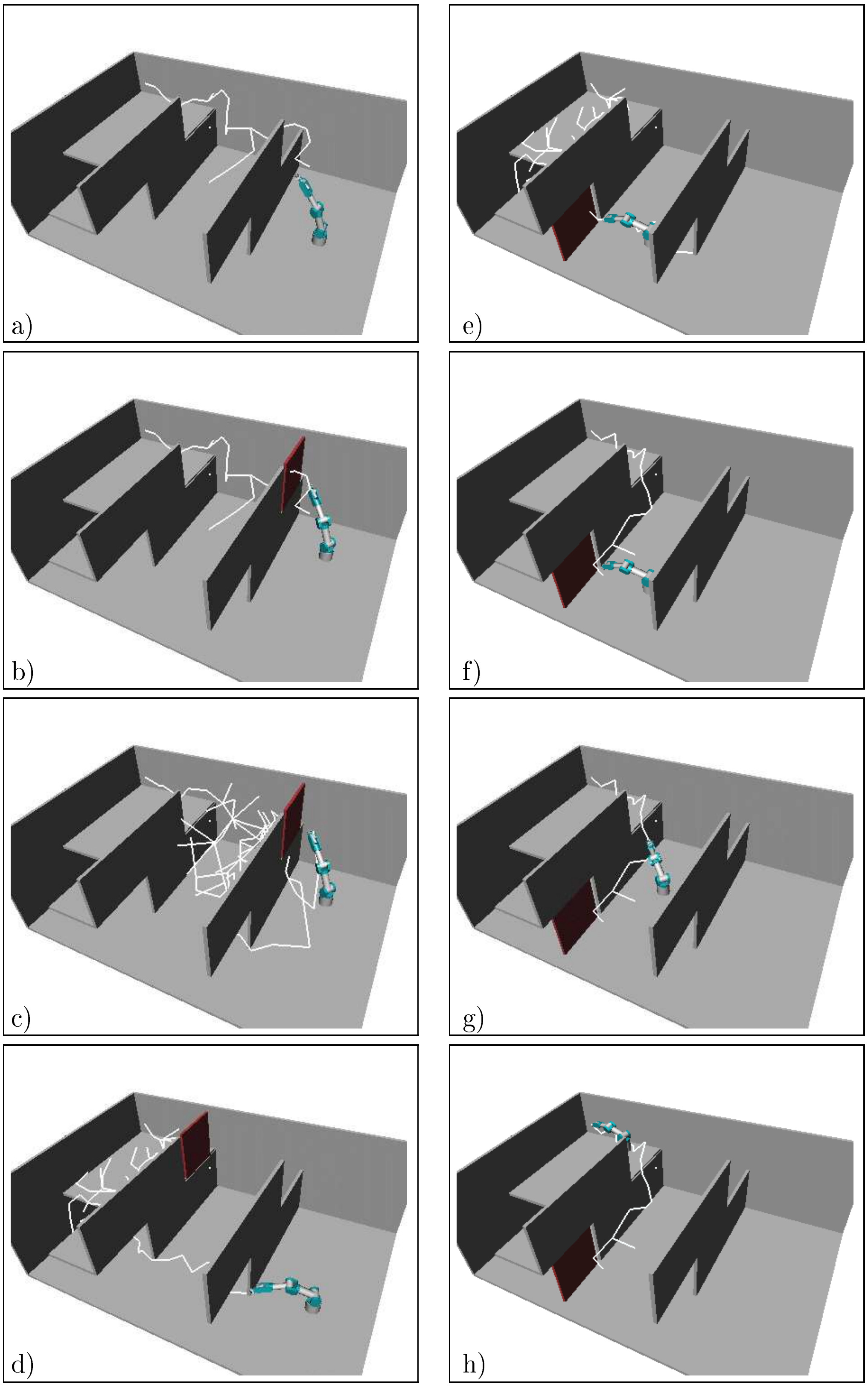

Figure 5: Real-time planning in a dynamic environment.

5 Future Work

This paper presented decomposition-based motion planning as a general framework for addressing a certain class of planning problems in real time. The experimental application of this framework to a particular planning task was also presented. Quite naturally, the introduction of a new framework leaves a lot of room for future investigation of various aspects associated with that framework.

Decomposition-based planning can be applied to planning problems different from the one presented here. In the context of a different planning problem the relevance of choosing the right link between the two subproblems becomes apparent. The work presented in this paper uses a volume in Euclidean space; other planning problems might require an area in the plane or a four-dimensional volume in space-time ($\mathbb{R}^3 \times t$). The algorithms used to solve the subproblems P_1 and P_2 would vary depending on the link between them and the particular planning problem at hand. A "cookbook" for choosing the linking space between the subproblems P_1 and P_2 and for choosing algorithms to solve the subproblems depending the given planning problem would be very valuable.

A more detailed characterization of planning problems lending themselves to the application of decomposition-based planning needs to be given. The notion of δ-hardness was introduced for the subproblem P_1. A similar classification for the problem of overlapping volumes for different homotopic equivalence classes arising in P_2 would be valuable.

The notion of δ-completeness attempts to characterize the tradeoff of completeness for efficiency in subproblem P_1. Similar results for P_2 would be interesting extensions of the theoretical foundation of decomposition-based planning. Since the loss of completeness is a deliberate choice, a quantification of this loss as a function of parameters of the planning problem and the chosen algorithms could extend the applicability of this framework. A quantification can then lead to provably more complete algorithms, in particular for the subproblem P_2. Here, algorithms to overcome structural local minima can be used as a starting point for further investigation.

If a decomposition-based planning approach fails to find a solution path to a given planning problem, other more complete (and less efficient) planning approaches might still be able to determine a solution. To fully exploit the advantages of decomposition-based planning, it would be very interesting to integrate planners following this framework with other planners. The information gained during the failed planning process of the decomposition-based planner could be exploited by a probabilistic roadmap planner, for example.

There are a number of planning tasks requiring contact with the environment. Most of these tasks involve moving an obstacle, with operations like picking the object up and putting it down. It is conceivable that a planning problem can be partitioned into those parts that require contact with the environment, like picking up an object, and those where the robot moves in free space. The latter part could be solved by decomposition-based planning, while the motion in contact could be planned using different techniques. So rather than applying the decomposition-based paradigm in sequence with other paradigms, as mentioned in the previous paragraph, they could by used in conjunction.

Finally, when execution motion in contact with the environment, only one specific part of the robot makes contact. The decomposition-based planning paradigm could be extended to allow this part to move outside the tunnel T. Instead of avoiding collisions by ensuring that the robot is contained within the tunnel, such an extension can accomplish collision avoidance by maintaining the constraints imposed by a contact force with the environment directly using the dynamically decoupled control scheme presented in Section 3.2.2. Such an extension would significantly extend the applicability of decomposition-based motion planning.

6 Conclusion

To achieve real-time motion planning for robots with many degrees of freedom, a motion planning paradigm based on problem decomposition was proposed. The paradigm addresses planning problems in which a minimum clearance to obstacles can be guaranteed along the solution path. The overall planning problem is decomposed into two planning subtasks: capturing the connectivity of the free space in a low-dimensional space and planning for the degrees of freedom of the robot in its high-dimensional configuration space. The solution to the lower-dimensional problem is computed in such a manner that it can be used as a guide to efficiently solve the original planning problem. This allows decomposition-based planning to achieve real-time performance for robots with many degrees of freedom.

The key to real-time performance is the trading of completeness for efficiency. Using the solution to the low-dimensional problem to guide the solution of the original planning problem allows motion generation without the exhaustive search of a high-dimensional configuration space. In very tight and complicated workspaces, however, it is possible that no path will be found using this framework, even if one exists. In such cases a configuration space planner can be used to determine a path through the difficult section of the workspace. Since the generation of a plan in the proposed planning method only takes a fraction of a second, such a planner could be evoked after the proposed planner reports failure.

To characterize the kind of motion planning problem to which decomposition-based planning can be applied, the notion of δ-hardness was introduced. This can be seen as the theoretical foundation to develop a more thorough theory of completeness for decomposition-based planning.

This paper also presented a particular implementation of the decomposition-based planning framework, using an adaptive wavefront expansion algorithm to efficiently capture a volume of free space, which is in turn used to guide reactive motion control to find a trajectory for the robot, solving the original planning problem. Preliminary experimental results with an eleven degree-of-freedom robot were presented, verifying the real-time performance of the planner.

Acknowledgments

The authors would like to thank Attawith Sudsang and Christopher Holleman for their helpful insights and discussion in preparing this paper. Work on this paper has been supported in part by NSF IRI-970228, NSF CISE SA1728-21122N and by a Sloan Fellowship to Lydia Kavraki.

References

[1] Nancy Amato, B. Bayazid, L. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective*. AK Peters, 1998.

[2] Jérôme Barraquand and Jean-Claude Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649, 1991.

[3] Robert Bohlin and Lydia E. Kavraki. Path planning using lazy PRM. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 521–528, 2000.

[4] Oliver Brock. *Generating Robot Motion: The Integration of Planning and Execution*. PhD thesis, Stanford University, 1999.

[5] Oliver Brock and Oussama Khatib. Executing motion plans for robots with many degrees of freedom in dynamic environments. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 1–6, 1998.

[6] Oliver Brock and Oussama Khatib. High-speed navigation using the global dynamic window approach. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 341–346, 1999.

[7] Oliver Brock and Oussama Khatib. Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 550–555, 2000.

[8] John F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, 1988.

[9] John F. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 49–60, 1987.

[10] Wonyun Choi and Jean-Claude Latombe. A reactive architecture for planning and executing robot motions with incomplete knowledge. In *Proceedings of the International Conference on Intelligent Robots and Systems*, volume 1, pages 24–29, 1991.

[11] Leonidas J. Guibas, Christopher Holleman, and Lydia E. Kavraki. A probabilistic roadmap planner for flexible objects with a workspace medial-axis-based sampling approach. In *Proceedings of the International Conference on Intelligent Robots and Systems*, 1999.

[12] Christopher Holleman and Lydia E. Kavraki. A framework for using the workspace medial axis in PRM planners. In *ICRA*, volume 2, pages 1408–1413, 2000.

[13] David Hsu, Lydia E. Kavraki, Jean-Claude Latombe, Rajeev Motwani, and Stephen Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, pages 141–154. A K Peters, 1998.

[14] Lydia E. Kavraki, Peter Švestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[15] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[16] Oussama Khatib. A unified approach to motion and force control of robot manipulators: The operational space formulation. *International Journal of Robotics Research*, 3(1):43–53, 1987.

[17] D. E. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings of the International Conference on Robotics and Automation*, pages 1–6, 1987.

[18] James J. Kuffner. Goal-directed navigation for animated characters using real-time path planning and control. In *Proceedings of Captech'98*, 1998.

[19] James J. Kuffner, Jr. and Steven M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proceedings of the International Conference on Robotics and Automation*, volume 2, pages 995–1001, 2000.

[20] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, 1991.

[21] Jean-Paul Laumond and T. Siméon. Notes on visibility roadmaps and path planning. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2000.

[22] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report TR 98-11, Iowa State University, 1998.

[23] Peter Leven and Seth Hutchinson. Toward real-time path planning in changing environments. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2000.

[24] Tsai-Yen Li and Jean-Claude Latombe. On-line manipulation planning for two robot arms in a dynamic environment. *International Journal of Robotics Research*, 16(2):144–167, 1997.

[25] C. Pisula, K. Hoff, M. Lin, and D. Manocha. Randomized path planning for a rigid body based on hardware accelerated voronoi sampling. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2000.

[26] Sean Quinlan. Efficient distance computation between non-convex objects. In *Proceedings of the International Conference on Robotics and Automation*, volume 4, pages 3324–3329, 1994.

[27] Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *Proceedings of the International Conference on Robotics and Automation*, volume 2, pages 802–807, 1993.

[28] Libo Yang and Steven M. LaValle. A framework for planning feedback motion strategies based on a random neighborhood graph. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 544–549, 2000.

[29] Yong Yu and Kamal Gupta. An information theoretic approach to viewpoint planning for motion planning of eye-in-hand systems. In *Proceedings of the International Symposium on Industrial Robotics*, 2000.