Robotics: Science and Systems 2005
Cambridge, MA, USA, June 8-11, 2005

Efficient Motion Planning Based on Disassembly

Yuandong Yang Oliver Brock
Laboratory for Perceptual Robotics
Computer Science Department
University of Massachusetts Amherst

***Abstract*— Disassembly-based motion planning (DBMP) is a novel and efficient single-query, sampling-based motion planning approach for free-flying robots. Disassembly-based motion planning uses workspace information to determine the workspace volume of a potential solution path and uses this information to exclude large portions of configuration space from exploration. It also identifies the most constrained placements of the robot along the potential solution path. These placements are referred to as assemblies because they are highly constrained by the environment, much like parts in an assembly are constrained. The constraints limit the possible motions of the robot and thus can be exploited to further limit configuration space exploration. The use of these two sources of workspace information permits the solution of many practical problems with very limited configuration space exploration. This reduction in configuration space exploration results in performance improvements of several orders of magnitude, compared to state-of-the-art motion planning methods. For non-free-flying robots, disassembly-based motion planning performs at least as well as the sampling-based motion planning method it is based on.**

I. Introduction

We present a novel sampling-based, single-query motion planning method for free-flying robots. The proposed planner systematically reduces the amount of configuration space it has to explore to solve a particular motion planning problem. This is accomplished by an analysis of the workspace. The planner uncovers structure in the workspace and uses it to guide configuration space exploration. This structure is exploited in two ways:

a) Exploiting Workspace Connectivity: Since the workspace is of constant and low dimensionality, its connectivity can be computed very efficiently. We determine a volume of free work space through which the robot can plausibly move from the initial to the final configuration. The volume is referred to as *tunnel*. Plausibility is determined based on a simple geometric criterion. The exploration of the high-dimensional configuration space can then be restricted to those regions that represent configurations of the robot in which it intersects with the free workspace volume. This generally eliminates large portions of configuration space. Moreover, the workspace tunnel also includes information of the distribution of obstacles along the path. This information is essential in choosing appropriate sampling algorithms (below). The details of this procedure are presented in Section III-A.

b) Exploiting Geometric Constraints: The complexity of configuration space regions varies over the configuration space in most realistic applications. Because the variation is unknown a priori, this property gives rise to the narrow passage problem in sampling-based motion planning: the sampling density for the entire configuration space has to be chosen such that the most difficult narrow passage along the solution path can be solved. As a result, many computational resources are wasted in less difficult regions of the configuration space. Rather than being adversely affected by narrow passages, the proposed motion planning approach exploits them to accelerate motion planning. The environment imposes many geometric constraints on a collision-free configuration inside a narrow passage, thus we view such a configuration as an assembly. The geometric constraints of an assembly greatly facilitate disassembly, as they limit the possible directions of motion and thus the amount of configuration space that has to be explored. The proposed planner uses this insight to improve the efficiency of motion planning. The details of how narrow passages are found, how assemblies are determined, and how they are disassembled are given in Section III-B.

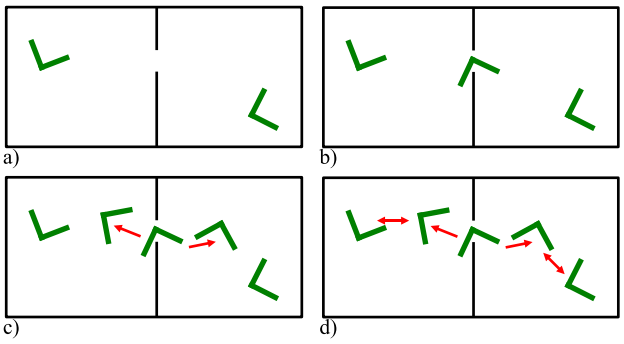


Fig. 1. Decomposition of a path planning problem into disassembly problems: a) initial and final configurations of a robot; b) an "assembled" state is determined based on information about the workspace connectivity; c) constraints imposed by the environment are used to solve disassembly problems; d) the motion planning problem is solved by connecting the disassembly sequences to the initial and final configurations in an open region.

By combining these two ways of exploiting workspace structure, we obtain a very efficient motion planner. We demonstrate the main ideas with an example as follows: given a motion planning problem shown in Figure 1a), workspace information is obtained and used to determine the spatial location of a narrow passage along a potential solution path. A collision-free configuration of the robot inside the narrow passage is obtained, as shown in Figure 1b). We consider this configuration to be an *assembled state* of the robot.

The corresponding disassembly problem can be solved much more efficiently than the assembly problem by exploiting the geometric constraints imposed by the environment (see Figure 1c). Finally, the disassembled states are connected to the initial and final configurations. This results in the overall solution path shown in Figure 1d). This can be done very efficiently, because the intermediate regions contain mostly free space. This principle also applies in environments with multiple narrow passages.

Effectively, the proposed planner divides the configuration space into three types of regions: *a)* the regions of configuration space that are not explored at all, because they represent configurations that do not intersect the workspace tunnel, thus irrelevant to the path query, *b)* difficult regions or narrow passages, i.e., regions in which the constraints imposed by workspace obstacles on collision-free configurations can be exploited to perform disassembly, and *c)* open regions that can be solved very efficiently. By performing this division, the proposed planner can distribute sampling densities accordingly and accelerate configuration space exploration significantly, yielding tremendous performance improvements over existing sampling-based path planners.

Performance improvements for the proposed planner are most pronounced for free-flying robots, because for these types of robots the configuration space can most effectively be divided into the three aforementioned categories. For other types of robots, the performance of decomposition-based motion planning will be equivalent to that of the underlying sampling-based motion planning approach. However, we believe that the principle of exploiting structure to render configuration space exploration is general and can lead to other efficient planners for the general motion planning problem [5].

II. Related Work

The original probabilistic roadmap (PRM) approach [12] gives rise to the narrow passage problem: paths through narrow passages in configuration space can either not be found, or the sampling density for the entire configuration space has to be prohibitively high. To alleviate this problem, some approaches perform sampling informed by workspace information. By producing samples in the proximity of the medial axis of the workspace, the likelihood of generating collision-free samples can be improved [9]. Information obtained in the workspace can also be exploited to locally adapt the sampling density [8], [14], [19], [21].

Decomposition-based motion planning methods [4], [8] decompose the motion planning problem into a low-dimensional and a high-dimensional subproblem. The low-dimensional motion planning problem can be solved efficiently in the workspace. The solution to this problem captures connectivity information in the workspace relevant to the high-dimensional planning problem. By exploiting this information, a solution to the overall motion planning problem can be computed efficiently. The method proposed in this paper differs from [8] in that it replaces the computation of the generalized Voronoi diagram with an efficient and localized sphere expansion; it further differs in that it can easily handle geometrically complex robots (since it avoids the "orientation" phase); and—most importantly—it differs in that it decomposes motion planning problems into *disassembly* problems. Generally, planners attempt to find paths *into* narrow passages. In contrast, disassembly-based planning *first* identifies narrow passages in the workspace to then find configuration space paths *out of* them, greatly facilitating the motion planning process. Whereas in [8] the focus is on exploiting workspace information in easy regions of the workspace, decomposition-based motion planning uses workspace information to solve the narrow passages of a motion planning problem. Furthermore, since decomposition-based motion planning is entirely based on sampling-based techniques, we can maintain probabilistic completeness guarantees.

A number of enhanced sampling strategies for multi-query motion planning have been proposed [1], [3], [7], [10], [16]. These methods exploit local configuration space properties to identify samples believed to contribute to a useful roadmap. Single-query planners guide the exploration of configuration space based on a particular motion planning query [2], [6], [11], [13], [18]. They employ heuristics to reduce the region of configuration space explored during planning.

III. Disassembly-Based Motion Planning

In this section, we describe the disassembly-based motion planning approach (DBMP). First, we explain how workspace information can be obtained efficiently. Then, we present the methods of using this information to reduce the amount of configuration space that must be explored to solve a given motion planning problem.

A. Obtaining Workspace Information

A robot sweeps out a workspace volume as it moves from an initial to the final configuration. If this workspace volume were known, the exploration of configuration space could be restricted to the subset of configurations at which the robot is contained within this volume. Decomposition-based approaches to motion planning compute an approximation of this workspace volume, called a workspace tunnel [4]. The exploration of configuration space can then be restricted to those configurations for which the robot partially overlaps with the tunnel. The set of qualifying configurations represents a small subset of the overall configuration space and consequently exploration can be performed much more efficiently. Disassembly-based motion planning uses the same idea, and thus belongs to the family of decomposition-based motion planning approaches.

1) Sphere expansion: The workspace tunnel is computed using a sphere expansion algorithm. The details of this algorithm are given elsewhere [4]; here, we outline the general idea. The purpose of the sphere expansion algorithm is to determine a continuous workspace volume through which the robot might be able to move from its initial position to its final position. Intuitively, sphere expansion is a wavefront propagation algorithm with adaptive step sizes.

To compute a workspace tunnel, we use a wavefront of free space spheres with maximum radius. Initially, the largest sphere of free space centered at a reference point of the robot in its initial position is computed. The radius of this sphere is determined by the distance of the reference point to the closest obstacle. The surface of the sphere is sampled and maximal spheres of free space centered at the sample points are determined. If the size of a sphere does not allow the robot to move through it, it is not expanded further. The remaining spheres are kept in a priority queue, with the highest priority assigned to the sphere closest to the final position of the robot. This process is referred to as sphere expansion. Expansion of the highest-priority sphere is performed until a sphere contains the reference point of the robot in its final position. The priority assignments of the spheres determine the exploration pattern of the workspace. Different heuristics can be applied to attain the fastest exploration of the workspace.

During sphere expansion, the parent/child relationship of spheres is maintained. The resulting data structure is a tree of spheres. The root sphere contains the reference point of the robot in its initial position. The spheres of the tunnel represent a path from the root of the tree to the leaf containing the reference point in the final position of the robot. Figure 2 shows a resulting free space tunnel. The line segments connecting the centers of the spheres along the tunnel are referred to as the spine of the tunnel.

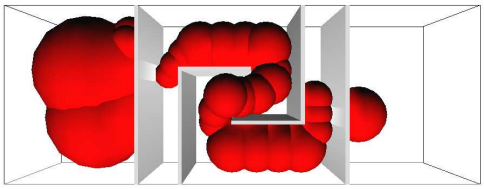


Fig. 2. Workspace tunnel computed using sphere expansion, connecting the initial position of the robot in the bottom, left part of the environment and the final position in the right part.

In contrast to other approaches that use workspace information [14], [19], [21], the sphere expansion provides information about workspace connectivity for a specific path query; exploration of the entire workspace is not necessary.

2) Exploring alternative tunnels: The sphere expansion method described above determines a workspace volume that connects the initial to the final configuration and has a certain minimum diameter. The requirement of minimum diameter is not sufficient to ensure that a particular tunnel also contains a valid solution to the planning problem. Should the planner at a later stage discover that no such path exists, an alternative tunnel has to be considered. Such alternative tunnels can be computed with a small modification to the sphere expansion algorithm described in the previous section.

If configuration space sampling identifies a particular region of the workspace tunnel as blocked, the corresponding spheres of the tunnel are marked and their children are removed from the queue. Sphere expansion then resumes normally until another workspace tunnel has been obtained (see Figure 3). This algorithm will explore alternative tunnels until the entire workspace has been examined.

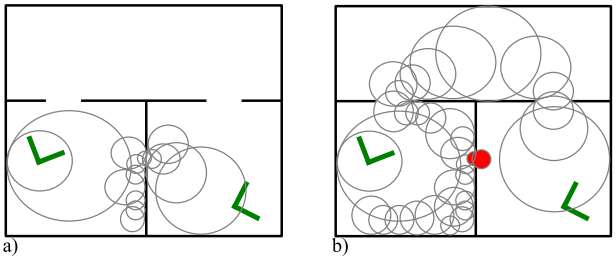


Fig. 3. For the initial tunnel (a) no assembly could be found. An alternative tunnel (b) is computed.

It should be noted that the computation of workspace information does not rely on any specific planning method and the connectivity information computed is helpful to *all* sampling-based motion planners. Once a candidate workspace tunnel has been computed, sampling is restricted to the tunnel. In the following, we will present a particular sampling-based motion planner based on disassemblies that exploits additional information obtained in the workspace.

B. Solving Narrow Passages as Disassemblies

So far, we have determined a workspace tunnel using sphere expansion. We now describe how this tunnel can be used to solve the corresponding motion planning problem efficiently.

1) Finding narrow passages: An assembly represents a placement of the robot inside a narrow passage in the work space. To find assemblies, narrow sections of the computed tunnel are considered. A section is narrow, if the radius of spheres in this section is below a threshold. This threshold is a parameter of the proposed approach. It can easily be estimated based on the geometry of the robot [4].

Once narrow passages are identified, local sphere expansion is performed to improve the understanding of the workspace nearby. In the implementation, sphere expansion explores the area reached by the robot end-effector with the base fixed in the smallest sphere in the narrow region. To divide the motion planning problem into disassembly problems, the most difficult regions of the tunnel must be identified. This can be accomplished by using the watershed algorithm applied to the union of spheres contained in the tunnel and the additional spheres obtained in the additional exploration [19], [20]. The result will be a set of spheres representing each narrow passage, labeled by the narrow passage they belong to. Spheres between two adjacent narrow passages are also labeled as easy regions distinctly, as are the spheres connecting the initial and the final position to the first and last narrow passage along the tunnel.

An example of such a labeling is given in Figure 4. Figure 4 a) shows the initial and final position of the robot and the environment with a narrow passage. Figure 4 b) illustrates the tunnel computed by sphere expansion. The spheres obtained

by the local expansion and the labeling determined by the watershed algorithm are shown in Figure 4 c).

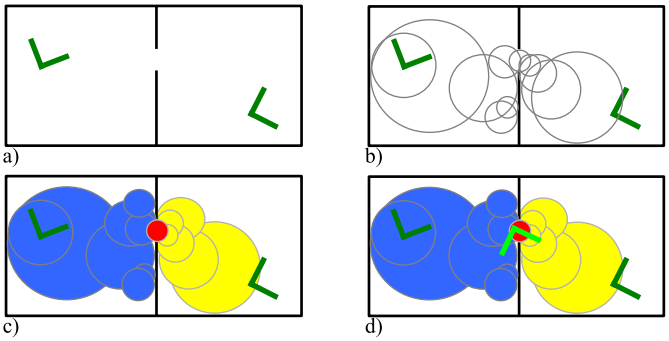


Fig. 4. Decomposition of the motion planning problem based on assemblies: a) the initial and final position of a robot; b) workspace tunnel connecting the initial and final positions; c) spheres are added to improve workspace understanding around narrow passages, spheres are labeled as the narrow passage and open regions (distinguished with different colors in the figure); d) the assembled state of the robot decomposes the motion planning into two subproblems: moving from the assembled state to the initial and final configurations.

2) Finding an assembly: An assembly is found by sampling robot configurations in the proximity of the narrow passage. Uniform sampling is adequate in this situation, because the entire region has roughly uniform complexity. To generate a sample inside the narrow passage for free-flying robots, a value for the rotational degrees of freedom of the robot is generated uniformly at random. In addition, a reference point on the robot and a random point inside the volume of the narrow passage are chosen. The translational component of the robot's configuration is determined such that these two points coincide.

Since an assembly should represent a highly constrained, collision-free placement of the robot, we require that the robot be either completely contained inside the narrow passage or can reach across it. In the latter case, we require that two points of the robot be contained in spheres with different labels outside the narrow region. An example of such a placement can be seen in Figure 4 d). Collision-free samples that do not fulfill this criterion are rejected. If more than one assembly is discovered after a certain number of samples, we choose a set of connected assemblies randomly to disassemble. If no assembly can be found, a new tunnel has to be computed.

3) Performing disassembly: So far only a very small fraction of the configuration space has been explored by uniform sampling. These regions correspond to narrow passages along a workspace tunnel connecting the initial and the final configuration. This computational expense is necessary, however, to solve the most difficult parts of the motion planning problem. The resulting assemblies give us much information about solutions for the remaining disassembly problems. We will use the configuration of the assembly to bias our sampling scheme for disassembly. The sampling scheme is motivated by two insights:

1) the translational component of the disassembly motion should be biased to move the robot out of the narrow passage and into an open region, and
2) due to the constraints imposed by the environment, only small incremental motions should be attempted.

Pseudo code for the proposed sampling procedure for diffusion-based disassembly is shown in Figure 5. Given a set A of connected assembled placements of the robot inside the narrow passage, this procedure builds a roadmap for each disassembly subproblem. Since assemblies are strictly constrained by the environment, samples are obtained by small perturbations of milestones already presented in the roadmap. This effectively uses the information contained in the initial assembly and subsequently the entire roadmap to reduce configuration space exploration by limiting samples in the regions likely containing collision-free placements of the robot. While the rotational component is perturbed randomly, the translational component of the configuration is perturbed with a bias towards the open regions of workspace represented by the tunnel. This bias exploits workspace information to further reduce configuration space exploration, since promising directions for translational motions are sampled more densely.

DISASSEMBLE (assemblies A, tunnel T)
  initialize roadmap R to contain assemblies A
  while robot has not left narrow passage
    randomly select milestone m from R
    randomly select direction d towards open region in T
    perturb translation of m biased by d to obtain m'
    perturb rotational components of m' to obtain m''
    if m'' is collision free
      if $r \in R$ exists so that m'' can be connected to r
        insert m'' and edge (r, m'') into R

Fig. 5. Pseudo code for disassembly; A represents a list of assembled states, T refers to the workspace tunnel, R designates a roadmap; m, m', m'', r are configurations.

By selecting configurations for perturbation uniformly at random in the roadmap R, rather than in a biased fashion as in other approaches [11], [13], we effectively achieve a bias towards promising regions of configuration space. This can be seen as follows. Since the configuration space region under consideration represents a narrow passage, collision-free samples will be rare. Once a collision-free sample has been found, its neighborhood is likely to contain additional free placements of the robot. This probability increases, as configurations move away from the assembled state, effectively introducing a bias from constrained regions towards less constrained regions. This bias is desirable for disassembly.

An assembly is considered to be disassembled, once the robot has been removed from the narrow section of the tunnel. Consecutive disassembled configurations along the tunnel are connected using the traditional PRM framework with uniform sampling [12]. Only samples in proximity to the relevant section of the workspace are retained. Consequently, the PRM framework is only applied to a small and open region of the configuration space and a small roadmap suffice to find a

solution.

C. Connecting Disassemblies

By concatenating disassembly sequences obtained by the proposed sampling scheme and intermediate paths obtained using a localized PRM planner, a solution to the initial motion planning problem can be determined. A localized PRM planner uses the sampling scheme described in Section III-B.2, instead of a uniform sampling scheme. This restricts the generation of samples to the region of interest by ensuring that the robot intersects with the relevant workspace region.

D. Probabilistic Completeness

Due to space limitations we can only provide high-level arguments for the probabilistic completeness of disassembly-based motion planning. Ultimately, we defer to the completeness proof for probabilistic roadmap methods with uniform sampling [17] and show that the algorithmic differences between the two motion planning methods do not affect the proof.

Our argument requires that all applied sampling methods are probabilistically complete; uniform sampling fulfills this requirement [17]. The implementation described in Section III uses uniform sampling to find assemblies. Uniform sampling is also used in easy regions. To ensure probabilistic completeness for uniform sampling, we can show that restricting sampling to workspace tunnels does not affect completeness.

For disassemblies in difficult regions, however, a diffusion-based technique is employed. To show completeness for the disassembly step, we require that a certain fraction of the samples placed during disassembly are placed uniformly at random in the disassembly regions; we then use the same argument as for uniform sampling. (In practice, we found that the sampling method described in Section III-B.3 is much more efficient than uniform sampling. We did not observe any problem instance solvable by uniform sampling, but not by using the disassembly method.)

To argue probabilistic completeness, we have to make an additional modification to the motion planner presented in Section III. This modification addresses the special case when the robot has to perform a motion for the purpose of re-orienting itself. Such a situation can arise when the initial and final configuration lie in the same difficult region. This difficult region can contain a valid workspace tunnel, but might not allow the robot to perform the required motion to assume the goal configuration. However, a solution path may still exist. Such a solution path would lead the robot from the difficult region into an easy region for re-orientation and then back into the difficult region. The workspace-based exploration of configuration space would not generate such a path, since it only generates samples along non-overlapping workspace tunnels connecting the initial and final placement of the robot. To find such a solution path, the workspace exploration will, after all other workspace tunnels have been rejected, generate a tunnel consisting of the entire workspace (see Figure 6). At this stage, the proposed planner would still benefit from the different sampling densities in different regions of the configuration space. The advantage gained from the restriction of sampling to a particular configuration space regions would be lost. Effectively, the planner is transformed from a single-query to a multi-query planner.

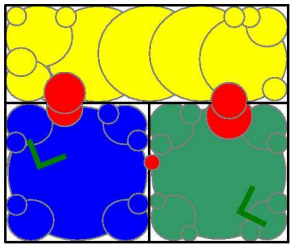


Fig. 6. A workspace tunnel resulting from a sphere expansion in the entire workspace.

Given these modifications to the algorithm, we rely on the completeness proof for probabilistic roadmap methods with uniform sampling [17] to argue that decomposition-based motion planning is probabilistically complete.

IV. Experimental Results

To evaluate the proposed planner, we compare it with six prevailing sampling-based planning methods: probabilistic roadmap (PRM) method with uniform sampling [12], PRM with Gaussian sampling [3], PRM with the bridge test [10], visibility-based PRM [16], rapidly-exploring random trees (RRTConnect) [13], and lazyPRM [2]. RRTConnect and lazyPRM can be viewed as the most commonly used single-query approaches. Since disassembly-based motion planning (DBMP) is a single-query method, these planners are most appropriate for comparison. We also compare the performance of disassembly-based motion planning with common multi-query planners. Our experiments indicate that in some environments these multi-query planners outperform the single-query planners (RRT and lazyPRM), but not disassembly-based motion planning. An additional comparison of the proposed method with a sampling-based motion planner using an approximated medial axis (aMAPRM [21]) can be found in [22].

The implementation used for the experiments is based on the Motion Strategy Library (MSL) [15]. This library contains implementations of PRM with uniform sampling and RRTConnect. Other motion planners used in the experimental evaluation were integrated into MSL.

The experimental environments used for the evaluation of disassembly-based motion planning are shown in Figure 7. The two left-most environments in Figure 7 contain two chambers connected by a narrow passage. The initial and goal configurations are located in different chambers. The two environments only differ in the width of the narrow passage. These environments are used to demonstrate that, in contrast to the other planners, the performance of disassembly-based motion planning is only marginally affected by the width of the narrow passage.

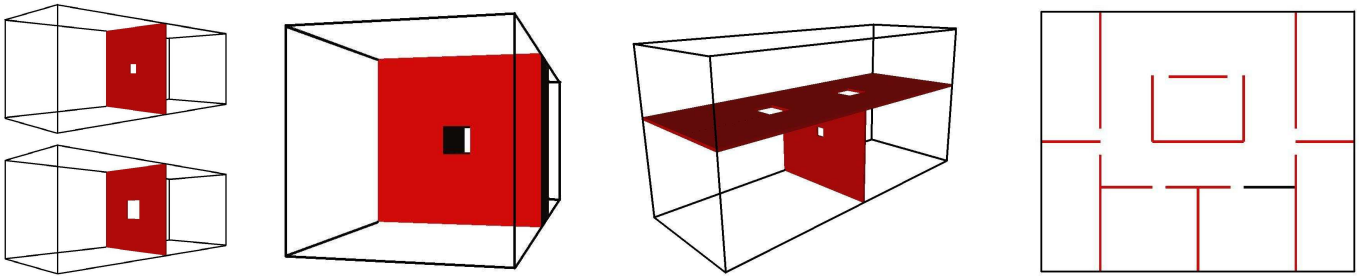


Fig. 7. Experimental environments: In the first environment on the left is bounded by a box ($12m \times 4.5m \times 4.5m$) and contains two parts of free space connected by a narrow passage (we consider two passages of different size: $0.1m \times 0.5m \times 0.5m$ and $0.1m \times 1.0m \times 1.0m$). The second environment is bounded by a box ($12m \times 4.5m \times 4.5m$) and the two parts of free space are connected by a long tunnel ($2.5m \times 1.0m \times 1.0m$); this tunnel is longer than the PA10 robot, which has a total length of 1.3m. The third environment is bounded by a box ($12m \times 7.5m \times 4.5m$) and the tunnel connecting the lower two chambers has a size of $0.1m \times 0.375m \times 0.375m$; both passages connecting the upper chamber to the lower chambers have a size of $0.1m \times 1m \times 1m$. The fourth environment is bounded by a box ($12m \times 10m \times 4.35m$) and contains several rooms. The doors have width of $0.5m$.

The second picture from the left in Figure 7 shows an environment in which the length of the narrow passage exceeds the length of the robot. This experiment illustrates that narrow passages can be solved efficiently, irrespective of their sizes.

The next environment contains three chambers, of which the two bottom chambers are connected by a passage that is very narrow for the robot. DBMP identifies this tunnel as an area containing a possible solution path. When the process of finding an assembly fails, the alternative tunnel is found and a solution path along the corresponding workspace volume is computed. This illustrates DBPM's ability to handle the failure of finding an assembly or the appropriate disassembly.

The right-most environment in Figure 7 represents an office building with hallways and offices. A projection of the model into the plane is shown. The robot moves between the rooms, connected by the hallway. This experimental environment illustrates how large portions of the configuration space can be excluded from motion planning based on workspace connectivity.

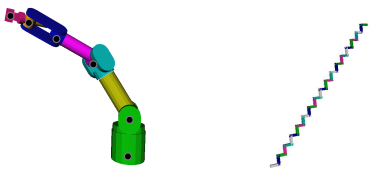


Fig. 8. Robots used in the experiments. Left: free-flying Mitsubishi PA-10 with 13 degrees of freedom, total length $1.317m$; Right: free flying 102 degree-of-freedom robot, composed of 33 links ($0.1mm \times 0.0125mm \times 0.0125mm$, total length $2.9m$) with ball-joints.

We ran experiments with a thirteen degree-of-freedom, free-flying PA10 robot and a 102 degree-of-freedom robot (Figure 8). Experimental results are reported in Table I. Our method outperforms other planners not only with respect to overall planning time, but also with respect to the size of the resulting roadmap. As discussed in Section III, DBMP spends most of the computational resources finding assemblies and disassembling them, i.e., solving those parts of the planning problem that are considered difficult. Only very limited resources suffice to connect disassembled states through easy regions. This indicates DBMP distributes the computational resources according to the difficulties of the regions. As a result, the roadmap contains most samples in difficult regions, where a denser sampling is required to capture the connectivity of the restricted free space. In open regions very few samples suffice to capture connectivity.

The experiments with the 102 degree-of-freedom robot demonstrate the ability of DBMP to solve planning problems in high dimensional configuration spaces. In the office environment, more time is spent connecting paths in the easy regions because the rooms and the corridor are small with respect to its size. In [22], we also demonstrate the effectiveness of DBMP applied to rigid body robots.

PRM with uniform sampling, PRM with Gaussian sampling, PRM with bridge test, and visibility-based PRM are multi-query planners. They explore the entire configuration space. It therefore is not surprising that disassembly-based motion planning can outperform these methods. It is noteworthy, however, that these multi-query methods in some environments outperform the single-query methods RRTConnect [13] and lazyPRM [2]. We offer the following explanation: RRTConnect prefers to drive configuration space exploration towards unexplored and open regions. It thus has difficulties to find paths from open regions into narrow passages. It also suffers from the translation-rotation discrepancy problem [7]. LazyPRM, on the other hand, biases exploration of configuration space based on proximity to the straight line connecting initial and final configuration of the planning problem. The experimental environments used in this paper contain a narrow passage that has to be traversed to solve the planning problem. If this narrow passage is "far away" from the straight line, lazyPRM is forced to exhaustively explore large regions of the configuration space.

Finally, to illustrate the sparseness of roadmaps resulting from disassembly-based motion planning, we show the samples of a successful roadmap in Figure 9. All milestones of the roadmap are along the workspace tunnel and most of them are close to the narrow passages.

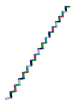
Robot	Workspace Computation			Path Planning							Total	
	Environment	Planner	$T_s(s)$	N_v	N_e	N_c	T_a	T_d	T_e	$T_c(s)$	$T(s)$	Factor
		PRM	N/A	18259	36516	29849130	N/A	N/A	N/A	6962.02	6962.02	5481.9
		Gaussian	N/A	9075	18146	2968297	N/A	N/A	N/A	1993.08	1993.08	1569.4
		BridgeTest	N/A	2361	4703	1997788	N/A	N/A	N/A	983.74	983.74	774.6
		Visibility	N/A	53	103	2614581	N/A	N/A	N/A	1526.12	1526.12	1201.7
		RRTConnect	N/A	12273	12272	804802	N/A	N/A	N/A	1174.95	1174.95	925.2
		lazyPRM	N/A	no path found after 20,000 seconds								>15748.0
		DBMP	**0.08**	**14**	**26**	**1444**	**0.27**	**0.92**	**0.00**	**1.19**	**1.27**	1.0
		PRM	N/A	1441	2878	269144	N/A	N/A	N/A	148.02	148.02	208.5
		Gaussian	N/A	335	662	66086	N/A	N/A	N/A	40.20	40.20	56.6
		BridgeTest	N/A	331	633	270557	N/A	N/A	N/A	130.57	130.57	183.9
		Visibility	N/A	25	49	222557	N/A	N/A	N/A	122.61	122.61	172.7
		RRTConnect	N/A	80	79	28591	N/A	N/A	N/A	17.75	17.75	25.0
		lazyPRM	N/A	7816	373357	91228	N/A	N/A	N/A	3128.96	3128.96	4407.0
		DBMP	**0.08**	**8**	**14**	**932**	**0.13**	**0.50**	**0.00**	**0.63**	**0.71**	1.0
		PRM	N/A	no path found after 20,000 seconds								>3738.3
		Gaussian	N/A	1945	3854	492442	N/A	N/A	N/A	381.02	381.02	71.2
		BridgeTest	N/A	473	869	352156	N/A	N/A	N/A	164.76	164.76	30.8
		Visibility	N/A	64	107	331750	N/A	N/A	N/A	199.13	199.13	37.2
		RRTConnect	N/A	8471	8470	657739	N/A	N/A	N/A	700.90	700.90	131.0
		lazyPRM	N/A	no path found after 20,000 seconds								>3738.3
		DBMP	**3.77**	**13**	**24**	**1487**	**0.08**	**1.50**	**0.00**	**1.58**	**5.35**	1.0
		PRM	N/A	2798	5590	3440460	N/A	N/A	N/A	2029.61	2029.61	253.7
		Gaussian	N/A	894	1772	200279	N/A	N/A	N/A	136.97	136.97	17.1
		BridgeTest	N/A	923	1781	1020581	N/A	N/A	N/A	466.78	466.78	58.3
		Visibility	N/A	80	153	1251411	N/A	N/A	N/A	684.25	684.25	85.5
		RRTConnect	N/A	5245	5244	501984	N/A	N/A	N/A	383.24	383.24	47.9
		lazyPRM	N/A	no path found after 20,000 seconds								>2500.0
		DBMP	**1.36**	**43**	**84**	**6773**	**0.45**	**4.36**	**1.83**	**6.64**	**8.00**	1.0
		PRM	N/A	2580	5510	6485085	N/A	N/A	N/A	4082.56	4082.56	249.1
		Gaussian	N/A	1028	1993	209273	N/A	N/A	N/A	170.60	170.60	10.4
		BridgeTest	N/A	589	1004	427681	N/A	N/A	N/A	237.94	237.94	14.5
		Visibility	N/A	165	307	436722	N/A	N/A	N/A	321.15	321.15	19.6
		RRTConnect	N/A	5934	5933	550687	N/A	N/A	N/A	479.08	479.08	29.2
		lazyPRM	N/A	no path found after 20,000 seconds								>1220.3
		DBMP	**10.80**	**39**	**76**	**4261**	**1.64**	**1.97**	**1.99**	**5.59**	**16.39**	1.0
		PRM	N/A	1734	3456	370109	N/A	N/A	N/A	3137.48	3137.48	330.3
		Gaussian	N/A	3977	7425	1268621	N/A	N/A	N/A	11402.24	11402.24	1200.2
		BridgeTest	N/A	2309	4056	1036890	N/A	N/A	N/A	7508.79	7508.79	790.4
		Visibility	N/A	41	64	317935	N/A	N/A	N/A	2590.49	2590.49	272.7
		RRTConnect	N/A	no path found after 20,000 seconds								>2105.3
		lazyPRM	N/A	no path found after 20,000 seconds								>2105.3
		DBMP	**2.00**	**7**	**12**	**1459**	**2.03**	**5.47**	**0.00**	**7.50**	**9.50**	1.0
		PRM	N/A	3299	6514	1229858	N/A	N/A	N/A	10612.4	10612.4	375.4
		Gaussian	N/A	1335	2358	343108	N/A	N/A	N/A	3264.92	3264.92	115.5
		BridgeTest	N/A	1563	2509	658825	N/A	N/A	N/A	5345.59	5345.59	189.1
		Visibility	N/A	109	206	642390	N/A	N/A	N/A	5577.04	5577.04	197.3
		RRTConnect	N/A	no path found after 20,000 seconds								>707.5
		lazyPRM	N/A	no path found after 20,000 seconds								>707.5
		DBMP	**7.44**	**23**	**44**	**2735**	**3.02**	**17.82**	**0.00**	**20.83**	**28.27**	1.0
		PRM	N/A	1407	2656	535469	N/A	N/A	N/A	6239.24	6239.24	47.6
		Gaussian	N/A	1365	2309	324785	N/A	N/A	N/A	3612.45	3612.45	27.6
		BridgeTest	N/A	1134	1618	356464	N/A	N/A	N/A	3545.02	3545.02	27.0
		Visibility	N/A	182	324	512168	N/A	N/A	N/A	5381.38	5381.38	41.1
		RRTConnect	N/A	no path found after 20,000 seconds								>152.6
		lazyPRM	N/A	no path found after 20,000 seconds								>152.6
		DBMP	**34.70**	**63**	**122**	**8082**	**0.73**	**53.08**	**42.55**	**96.36**	**131.07**	1.0

TABLE I

Experimental Results

Comparison of a PRM planner with uniform sampling (PRM), a PRM planner with Gaussian sampling (Gaussian), a PRM planner with bridge test BridgeTest, a visibility-based PRM (Visibility), an RRTConnect planner, a lazyPRM planner and a disassembly-based motion planning (DBMP). T_s represents the time to compute the workspace connectivity information; N_v denotes the number of vertices in the roadmap, N_e refers to the number of edges in the roadmap, N_c specifies the total number of collision checks, T_a is the time to find assemblies in the narrow passage(s), T_d represents the time to disassemble the robot from assemblies, T_e denotes the time to connect configurations in easy regions, T_c gives the duration of roadmap construction, and T is the add-up time consumption. Factor indicates the time consumption ratio between other planners and DBMP. All times are averaged over ten runs, given in seconds. The experiments were performed on a PentiumIV 3.2GHz PC with 1GB RAM and a 64MB DDR Radeon 300 graphics card.

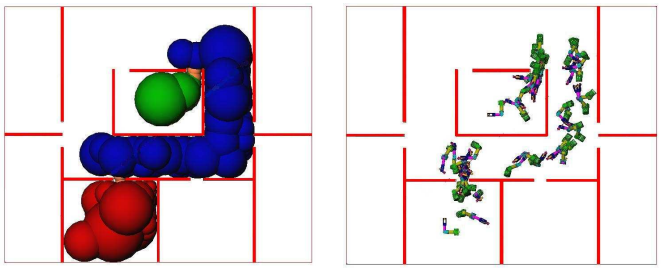


Fig. 9. Sample placement for disassembly-based motion planning: The planning problem requires the PA10 robot to move from one room to another in the floor environment. Left: the workspace tunnel connecting the initial and goal positions of the robot. Right: milestones of the resulting roadmap projected into the workspace. There are clusters of milestones inside the narrow passages. This indicates that most computation time is spent solving the difficult portions of the motion planning problem.

V. Conclusion

Disassembly-based motion planning is a single-query, sampling-based motion planning method for free-flying robots that outperforms existing motion planning techniques by several orders of magnitude for many realistic scenarios. We present extensive experimental evidence in support of this argument, comparing the performance of disassembly-based motion planning with several state-of-the-art sampling-based motion planners. For stationary robots the planner is capable of achieving performance improvements and will always perform at least as well as the underlying sampling-based motion planning approach.

Disassembly-based motion planning derives its efficiency from the exploitation of workspace information. This information is used to identify configuration space regions that are likely to contain the solution path. By restricting exploration to these regions, only a small fraction of configuration space need to be explored, significantly increasing the efficiency of the motion planning process.

To restrict configuration space exploration, two distinct types of workspace information are exploited. Connectivity information from the workspace is used to identify a tunnel of free workspace likely to contain a solution path. Configuration space exploration can then be restricted to configurations in which the robot intersects this workspace tunnel. Note that this aspect of the proposed planner can be applied as a pre-processing step to any sampling-based motion planner to significantly improve its performance.

Workspace information is further used to identify narrow passages. A collision-free placement of the robot inside a narrow passage is considered an *assembly*, because geometric constraints of the environment limit the motions the robot can perform. The constraints imposed by the environment can in turn be exploited to limit configuration space exploration, resulting in further performance improvements.

References

[1] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 155–168, 1998.

[2] R. Bohlin and L. Kavraki. Path planning using Lazy PRM. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 521–528, 2000.

[3] V. Boor, M. H. Overmars, and A. F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1018–1023, 1999.

[4] O. Brock and L. Kavraki. Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces. In *Proceedings of the International Conference on Robotics and Automation*, volume 2, pages 1469–1474, 2001.

[5] B. Burns and O. Brock. Sampling-based motion planning using predictive models. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005.

[6] B. Burns and O. Brock. Single-query entropy-guided motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, Barcelona, Spain, 2005.

[7] E. Ferre and J.-P. Laumond. An iterative diffusion algorithm for part disassembly. In *International Conference on Robotics and Automation(ICRA)*, pages 3149–3154, April 2004.

[8] M. Foskey, M. Garber, M. C. Lin, and D. Manocha. A Voronoi-based hybrid planner. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 55–60, 2001.

[9] C. Holleman and L. E. Kavraki. A framework for using the workspace medial axis in PRM planners. In *Proceedings of the International Conference on Robotics and Automation*, pages 1408–1413, 2000.

[10] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2003.

[11] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications*, 9(4):495–512, 1999.

[12] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. Technical Report CS-TR-94-1519, Rice University, 1994.

[13] J. Kuffner and S. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 995–1001, 2000.

[14] H. Kurniawati and D. Hsu. Workspace importance sampling for probabilistic roadmap planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1618–1623, 2004.

[15] S. M. LaValle. http://msl.cs.uiuc.edu/msl.

[16] T. Simon, J.-P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6):477–493, 2000.

[17] P. Svestka. On probabilistic completeness and expected complexity of probabilistic path planning. Technical Report UU-CS-96-20, Dept. Comput. Sci., Utrecht Univ., 1996.

[18] D. Vallejo, I. Remmler, and N. M. Amato. An adaptive framework for single shot motion planning: A self-tuning system for rigid and articulated robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 21–26, Seoul, Korea, 2001.

[19] J. P. van den Berg and M. H. Overmars. Using workspace information as a guide to non-uniform sampling in probabilistic roadmap planners. In *International Conference on Robotics and Automation(ICRA)*, pages 453–460, April 2004.

[20] L. Vicent and P. Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:583–598, 1991.

[21] Y. Yang and O. Brock. Adapting the sampling distribution in PRM planners based on an approximated medial axis. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 4405–4410, 2004.

[22] Y. Yang and O. Brock. Viewing motion planning as disassmbly: A decomposition-based approach for non-stationary robots. Technical Report TR 04-108, Department of Computer Science, University of Massachusetts Amherst, December 2004.