Computer-Aided Design and Applications
© 2008 CAD Solutions, LLC
http://www.cadanda.com

D-Plan: Efficient Collision-Free Path Computation for Part Removal and Disassembly

Liangjun Zhang[1], Xin Huang[2], Young J. Kim[3] and Dinesh Manocha[4]

[1]University of North Carolina at Chapel Hill, zlj@cs.unc.edu
[2]University of North Carolina at Chapel Hill, huangxin@cs.unc.edu
[3]Ewha Womans University, Korea, kimy@ewha.ac.kr
[4]University of North Carolina at Chapel Hill, dm@cs.unc.edu
http://gamma.cs.unc.edu/d-plan

ABSTRACT

We present a novel approach to compute a collision-free path for part disassembly simulation and virtual prototyping of part removal. Our algorithm is based on sample-based motion planning that connects collision-free samples in the configuration space using local planning. In order to effectively handle the tight-fitting scenarios, we describe techniques to generate samples in narrow passages and efficient local planning algorithms to connect them with collision-free paths. Our approach is general, makes no assumption about model connectivity or object topology, and can handle polygon soup models that frequently arise in CAD applications. We highlight the performance on many challenging benchmarks including the Alpha puzzle, maintainability of the windscreen wiper motion, and disassembly of a seat from the interior of a car body.

Keywords: Part Disassembly Planning, Motion Planning, Virtual Prototyping
DOI: 10.3722/cadaps.2008.xxx-yyy

1. INTRODUCTION

The problems of assembly maintainability and mechanical part disassembly frequently arise in design and manufacturing applications. The manual generation of detailed disassembly or maintainability paths can be tedious and time consuming, particularly in environments prone to frequent design changes. The recent trend has been towards developing automated algorithmic solutions for such design problems that can automatically compute a collision-free, global path. These simulation technologies are increasingly used for virtual prototyping and PLM (product lifecycle management), where the goal is to provide efficient software solutions to problems that were traditionally solved using costly physical mockups.

The simulation of assembly maintainability attempts to remove a particular part from an assembly. Similarly, part disassembly simulation boils down to computing collision-free trajectories for objects through tight spaces or narrow passages. Many of these problems reduce to motion planning of robots, where collision-free paths need to be computed for rigid objects with six degrees of freedom (DOF) among stationary obstacles [4, 8, 11, 24, 30].

Motion planning has been extensively studied in robotics and related areas for more than three decades. At a broad level, prior approaches for rigid objects can be classified into exact algorithms based on algebraic formulation, local techniques that use potential fields or sample-based planning algorithms. In terms of CAD/CAM applications, most of practical planners are based on randomized sampling. These algorithms generate collision-free samples and attempt to connect these samples using local planning [15, 24]. The performance of these planners varies depending on how well they can capture the connectivity of the free space of the robot.

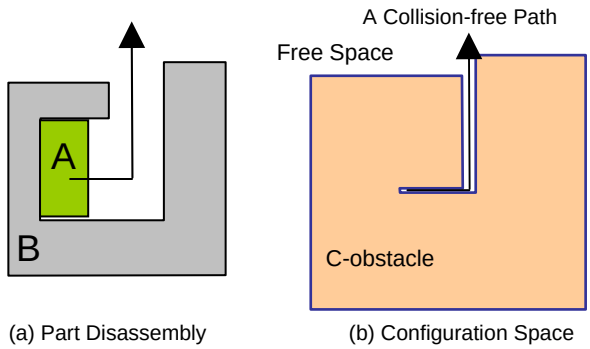


Fig. 1: Part Disassembly: (a) This figure highlights how a part disassembly problem can be reduced to motion planning for a rigid robot. In order to disassemble multiple parts, e.g. A and B, we treat A as a free-flying rigid robot and B as a static obstacle. (b) The problem of path planning of robots is formulated using the notion of configuration space. For the 2D robot A with two translational DOFs, its configuration space is 2 dimensional. The goal of computing a collision-free path is mapped to finding a path in robot's free space for a point robot, as shown here. For a general rigid robot in 3D with six DOFs, its configuration space is six dimensional.

A major challenge is to compute collision-free paths in narrow passages in the configuration space [2, 5, 14, 27]. The narrow passages are classified as regions, whose removal or perturbation can change the connectivity of the free space. Interestingly, the motion planning scenarios arising in part removal and part disassembly simulations are rather challenging in terms of narrow passages. Furthermore, the underlying models are complex and may be represented using thousands of polygons. Many times the models are given as polygon soup models with no connectivity or topology information. It is important for PLM applications that the motion planner should be able to handle such datasets automatically.

Main Results. In this paper, we present a general and fast motion planning algorithm, D-Plan, for part disassembly simulation. We use sample-based planning approaches and present two techniques to improve the performance. These include a fast retraction-based scheme to generate samples in narrow passages and near the boundary of the C-obstacle space. Secondly, we use constrained motion interpolation to effectively connect nearby samples by taking into account the non-collision constraint for the closest features between the robot and the obstacles.

We apply our approach to general, complex, polygon soup models. Such models are increasingly used in virtual prototyping and PLM, since many CAD systems import models generated from other sources, and sometimes the translators do not maintain the connectivity information. We present techniques to perform efficient contact query among polygon soup models and compute their closest features pairs. We further improve the performance of our planner by performing localized collision detection and exploit the spatial coherence between nearby queries in the configuration space.

We have tested the performance of D-Plan on difficult part removal and disassembly problems. These include the well-known Alpha puzzle [2], maintainability of the windscreen wiper motion with 15K triangles for the robot and 11K triangles for the obstacle, and disassembly of a seat (30K triangles) outside a car body (214K) described in [8].

Organization. The rest of the paper is organized as follows. In Section 2, we briefly survey prior work on part disassembly and motion planning. We give an overview of sample-based motion planning algorithms in Section 3 and highlight the issues in handling narrow passages. Section 4 describes our sample generation algorithm and the constrained interpolation technique. We present techniques to efficiently perform contact query and collision detection on general polygon soup models in Section 5. We highlight the performance of D-Plan on challenging benchmarks in Section 6.

2. Previous Work

In this section, we give a brief overview of prior work on part disassembly, sample-based motion planning and contact analysis.

2.1 Part Disassembly

Assembly and disassembly planning is a broad topic that has been extensively studied in CAD/CAM, virtual prototyping and motion planning. It mainly deals with the sequencing of the (dis)assembly operations of multiple parts to (dis)assemble a product [1, 21, 23, 32]. In this paper, we only focus on specific problems on part removal or maintainability study, where one needs to compute a collision-free path for a particular part that needs to be removed from an assembly. As shown in Fig. 1, part disassembly problem can be reduced to a motion planning problem, where the part to be extracted is treated as a robot and the rest of the assembly parts are treated as static obstacles. Some specialized motion planning algorithms based on random sampling and diffusion have been proposed for part disassembly [8, 9].

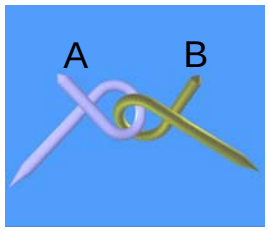
2.2 Sample-based Motion Planning

Sample-based motion planning, such as probabilistic roadmaps (PRM) [18] and rapidly-exploring random trees (RRT) [20] have been successfully used to solve high degree-of-freedom motion planning problems arising in different applications. However, current sample-based planners may not work well on part disassembly or removal problems due to inherent narrow passages. Many sampling strategies have been proposed to improve the performance in such cases [15]. Our approach for improving the performance in narrow passages is based on retraction-based methods [2, 5, 14, 25, 26, 29]. To connect the generated samples, most local planners use simple interpolation schemes, such as straight-line linear motion, screw motion or spherical linear motion and perform the collision checking [3, 19]. A few of sophisticated local planners have also been proposed [10, 12, 16], though their runtime overhead can be high.

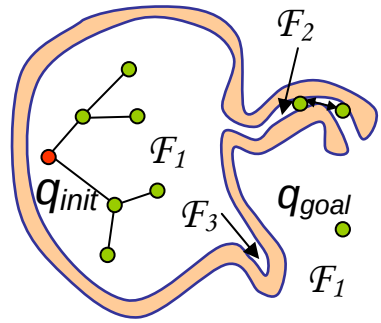
2.3 Contact Space Analysis

Contact space analysis is useful for part disassembly [6, 28]. It deals with modeling and searching the contact space, the collection of every possible configuration where the robot touches any obstacle. Due to the impracticality of explicitly representing the contact space exactly, some algorithms compute a high-level graph representation of the contact space based on geometric or algebraic techniques [33]. Other approaches have been proposed to randomly sample the contact space or generate compliant motion [7, 13, 17].

3. Collision-Free Path Computation



Alpha Puzzle Benchmark



Configuration Space

Fig. 2: Narrow Passages Arising in the Alpha Puzzle Benchmark: The goal is to separate the two intertwined alpha-shaped models. This problem reduces to the motion of the rigid robot A, while treating B as a static obstacle. The six dimensional free space of this problem is decomposed into three regions: $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ and this figure shows a 2-dimensional projection of those regions. $\mathcal{F}_2$ is a narrow passage. It is difficult for sample-based planners to perform sampling and local planning in such narrow passage due to its small volume.

Our goal is to compute collision-free paths for part removal and part disassembly simulation. This problem reduces to the motion of a rigid robot with six DOFs (Fig. 1), where the other parts of the assembly are treated as obstacles. We use the notion of configuration space, namely $\mathcal{C}$, where the robot is represented as a point, and the obstacles in the scene are mapped to the configuration space obstacles or C-obstacle, $\mathcal{O}$ [23]. The problem of finding a collision-free path for a robot can be mapped to computing a path for the point in the free space $\mathcal{F}=\mathcal{C}\setminus\mathcal{O}$. Most prior approaches can be classified based on how they represent or compute the free space. Specifically, we use sample-based planners that generate samples in $\mathcal{F}$ and attempt to connect those samples using a local planning algorithm. For the rest of the paper, a sample is classified as *in-colliding* if it lies in C-obstacle, *collision-free* if it lies in the free space, and *on-contact* if it lies on the contact space, the boundary of C-obstacle space.

3.1 Sampling in Narrow Passages

Many times the free space $\mathcal{F}$ for part disassembly or removal has the shape of a long thin tube [8], which geometrically corresponds to a narrow passage. The solution path tends to be very close to the boundary of the C-obstacle. In this case it is important that the sample-based planner generates a lot of samples near the C-obstacle space boundary. In part disassembly problems, the initial configuration may also lie in a narrow passage and some algorithms exploit this property to efficiently compute a collision-free path [8]. However, this kind of cases may not arise in part removal or other CAD benchmarks. This is illustrated for the well-known Alpha shape benchmark [2], which is widely regarded as a challenging benchmark for motion planning algorithms. The six dimensional space, $\mathcal{F}$, has narrow passages and we illustrate an approximate projection of $\mathcal{F}$ along the two dimensions in Fig. 2. For simplicity, we decompose the free space into three regions:

- $\mathcal{F}_1$: This region includes the initial and goal configurations. Based on the visibility characterization in sample-based approaches [15], most other configurations in $\mathcal{F}_1$ can be easily connected within $\mathcal{F}_1$.
- $\mathcal{F}_2$: This is a narrow passage and the solution path lies in this narrow passage (also shown in Fig. 2). As a result, any sample-based planner needs to generate a sufficient number of samples to capture the connectivity.
- $\mathcal{F}_3$: This is another narrow passage region, though it corresponds to a dead-end tunnel. Many sample-based planners would generate samples in this region, although may not be needed for the computation of the final path.

3.2 Local Planning in Narrow Passages

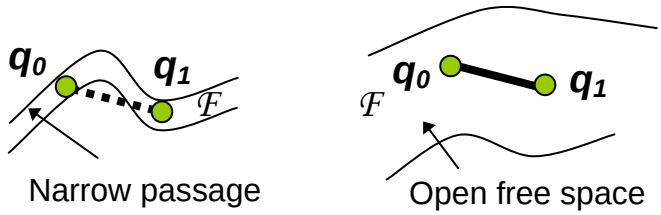


Fig. 3: Local Planning in Narrow Passage: Most local planning approaches use a straight-line or spherical linear interpolation motion to connect nearby samples in the free space. When dealing with a cluttered environment, such formulation tends to be less likely to connect samples with a collision-free path than in the open free space.

In addition to sample generation, such benchmarks with narrow passages are also challenging in terms of the local planning step. In general, it is often more difficult to perform local planning between samples in narrow passages than in the open free space as illustrated in Fig. 3. For simplicity, most local planning approaches use a straight-line or spherical linear interpolation to connect the samples. Such interpolation schemes are environment independent, which means that the generated motion only depends on the given samples, but is independent of the environment or the boundary of C-obstacle. Therefore, the generated motion may not be adaptive to the boundary. When dealing with a cluttered environment, it tends to be less likely to connect samples with a collision-free path than in open free space.

4. D-Plan

In this section, we give an overview of D-Plan, our path planning approach for part disassembly and removal. Using sample-based motion planning, our approach can efficiently compute a collision-free path for challenging part disassembly problems with narrow passages. As Fig. 4 shows, our approach includes two major components: a retraction-based sampling scheme for efficiently generating samples in narrow passages, and a local planner enhanced using constrained motion interpolation.

4.1 Retraction-based Sampling

Sample-based motion planning approaches explore the robot's free space by randomly sampling and building a tree (or roadmap) for computing a collision-free path as shown in Fig. 5. In order to handle part disassembly problems with narrow passages, one needs to generate sufficient non-colliding samples in these regions. We retract a randomly generated in-colliding sample to a more desirable location, i.e. to the boundary of the C-obstacle space. In this way, the in-colliding samples near the boundary of narrows passage can be retracted to narrow passages [14, 31, 35]. Next, we briefly describe our retraction algorithm.

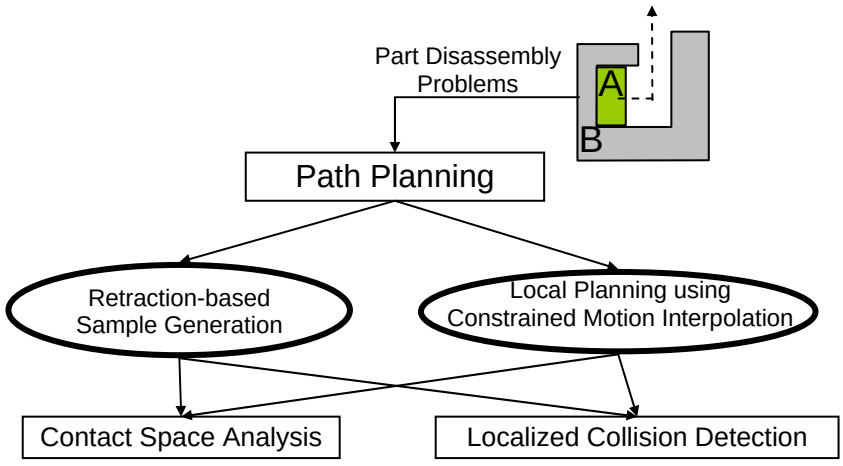


Fig. 4: Overview of D-Plan: A part disassembly problem reduces to the motion of a rigid robot. Using sample-based motion planning, our approach for collision-free path computation includes two new components: a retraction-based sample generation scheme and a local planner enhanced by constrained motion interpolation. Both the components perform contact queries and are accelerated by localized collision detection.

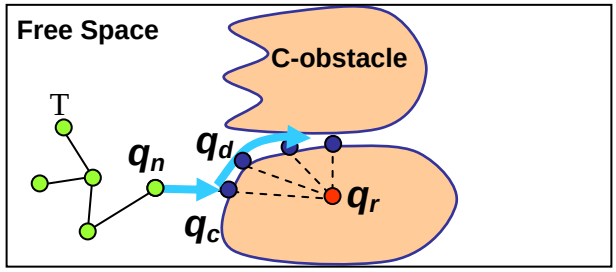


Fig. 5: Retraction-based Sampling Generation: To generate more non-colliding samples in the narrow passages, our approach retracts a randomly generated sample - q_r to a more desirable location incrementally. In this case, the sample q_r is incrementally retracted from q_n to q_c to q_d and so on.

We use an optimization-based retraction algorithm [35]. In order to retract a randomly generated in-colliding sample to a more desirable place, one common method is to retract it to the closest point of the boundary of C-obstacle. In practice, it is computationally prohibitive to compute an explicit representation of the boundary. Therefore, we may not be able to compute the closest point on the boundary exactly. In our formulation, we reduce the retraction computation to an optimization problem and iteratively refine the solution to compute a locally closest point. As shown in Fig. 5, to retract a given in-colliding sample q_r, our method starts with a non-colliding sample q_n (either collision-free or on the contact space) as the initial guess, which can be the nearest node from the current tree or the roadmap constructed for planning. The method then performs the following steps iteratively.

1. Project q_n onto the contact space to generate a sample q_c on the contact space;
2. Perform a contact query, i.e. computing the closest feature pairs between the robot at q_c and obstacles;
3. Search over the local contact space that is formed by the closest feature pairs and compute a new non-colliding sample q_d, which locally minimizes the distance to the sample q_r according to some distance metric;
4. Assign $q_n = q_d$ and go to Step 1.

These steps are iterated until the distance to q_r can not be further reduced or the maximum number of iterations have been reached. All collision-free and contact space samples generated during each iteration are collected and used to expand the tree or the roadmap for planning.

Our method for sample generation using the optimization-based retraction is general for any type of polygonal model. The method is also effective for sampling in narrow passages. This is because many in-colliding samples

near the boundary of C-obstacle can be retracted to the narrow passages, according to the analysis using Voronoi diagram [35]. We briefly describe some key techniques to implement this method. In Step 1, we use a binary search on the interpolating motion between q_n to q_r to compute a configuration q_c, at which the robot just touches the obstacles. In Section 5, we present a technique for computing the closest feature pairs among polygon soup models needed for Step 2. In Step 3, we first randomly generate samples over the local contact space. Since the CAD models are rigid, there are simple methods known to sample in the contact space [35]. We then find a non-colliding sample q_d from those random samples. In Section 5, we present our technique to accelerate this search by making use of the spatial coherence.

4.2 Local Planning using Constrained Motion Interpolation

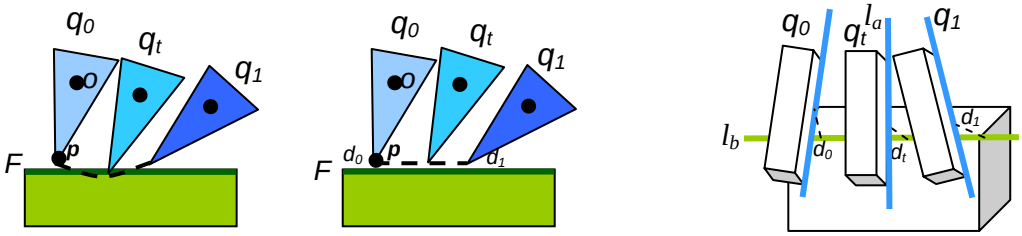


Fig. 6: Motion Interpolation between Configurations q_0 and q_1: (a) There is a collision at the configuration q_t if we use a linear interpolating motion; (b) The trajectory generated by our constrained motion interpolation algorithm. In this case, the sign of the distance between the vertex p and the plane containing the face F does not change along this path, and we can obtain a collision-free path. Overall, the use of constrained motion interpolation increases the probability of finding a collision-free path for the local planner. (c) A constrained interpolating motion for another case where the sign of the distance d_t between the highlighted lines l_a and l_b does not change.

We use a constraint-based motion interpolation algorithm to improve the performance of local planning for nearby samples [34]. Given two nearby samples, q_0 and q_1, a local planner often computes an interpolating motion and further checks whether it is collision-free. When local planning is performed near the contact space or in the narrow passages, the interpolating motion is more likely to collide with the C-obstacles. In order to compute a collision-free path, we first determine the closest feature pairs between the robot at the configuration q_0, as well as q_1 and the obstacles. Next, we try to ensure that each closest feature pair does not collide when the robot is transformed along the interpolating motion. Fig. 6 shows the simplest situation when there exists only a single pair of closest features reported, i.e. a vertex p of the robot and a face F of the obstacle. If we impose the distance constraint between this pair so that the sign of the distance between p and F does not change along the interpolating motion, we can guarantee there is no collision between p and F. Essentially, by taking into account local non-collision constraints among the feature pairs, our formulation increases the probability of finding a collision-free path for the local planner [34].

In order to compute a constrained motion, we use our contact query algorithm described in Section 5 to compute the closest feature pairs between the robot and the obstacles. In practice, multiple feature pairs could exist, when each feature pair realizes a local minimum distance. Also, the feature pairs reported at q_0 and q_1 can be different. In [34], we present our formulation in detail on how to handle the situation when there are two or three pairs of closest features. As a result, our constraint-based interpolation algorithm can deal with many situations in part disassembly planning with multiple pairs of locally closest features. Our algorithm can avoid the potential collisions between all these closest pairs (up to three) and tends to compute a path that is more likely to be collision-free.

5. General CAD Models

In Section 4, we gave an overview of our planning approach for part removal and disassembly, D-Plan. In this section, we present how to handle general polygon soup models for part disassembly planning. Specifically, many CAD systems import models generated from other sources, and sometimes the translators do not maintain the connectivity information. As a result, we need to ensure that D-Plan can easily handle general polygon soup models. We first present our efficient contact query method for such kind of models. Next, we utilize the spatial coherence between the nearby samples and use it to perform efficient collision detection.

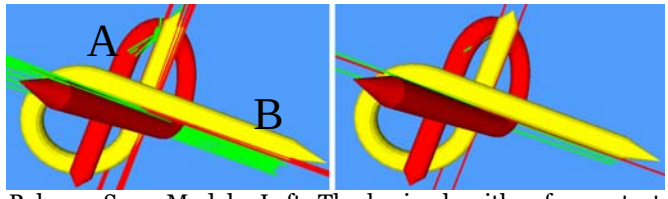


Fig. 7: Contact Query on Polygon Soup Models. Left: The basic algorithm for contact query is to determine all feature pairs between alpha-shaped models A and B, whose distances are less than a given tolerance. Many feature pairs are computed, while some of them are either duplicate or nearby. Right: We filter the duplicate or nearby pairs and compute the representative ones, i.e. the locally closest feature pairs of between the models. In this example, we obtain three representative pairs of features.

5.1 Contact Query on Polygon Soup Models

The basic algorithm for performing the contact query can be described as follows. We assume that each object corresponding to the robot or any obstacle is represented as a collection of triangles. We may or may not have any connectivity information among the triangles. Given two polygon soup models A and B, we build a bounding volume hierarchy (BVH) for each of them. By traversing both the BVHs, we can efficiently determine all the pairs of triangles between the two models, whose distance is less than some given tolerance κ_c. For each such triangle pair, we can further obtain the feature pair realizing the closest distance. In general, there are three types of feature pairs: (V,F), a vertex of A and a face of B, (F,V), a face of A and a vertex of B and an edge of A and an edge of B (E,E). We collect every such feature pair whose distance is less than κ_c, as Σ.

There could be many duplicate or nearby feature pairs in Σ, computed by the above basic algorithm, as shown in Fig. 7-(a). The duplicate feature pairs are generated because a feature in each model may be incident to more than one triangle. For example, for an (E, E) feature pair in Σ, if there are two triangles sharing an edge for each model, this feature pair is reported four times. Furthermore, the hierarchical algorithm described above will report many nearby feature pairs as long as their distances are smaller than the given tolerance κ_c.

The duplicate or nearby feature pairs need to be filtered from Σ. Otherwise, many unnecessary contact constraints are constructed, which can result in many extra samples during our sampling stage. In general, it is difficult to filter out duplicate or nearby feature pairs, since we do not have the connectivity information of the models. We use a simple heuristic for the purpose of filtering and computing the representative ones, i.e. the locally closest feature pairs between A and B. For each feature pair computed by the basic algorithm, we compute their witnessing points p_a and p_b in those features, i.e. the points which witness the closest distance between the two features. In order to determine whether a feature pair is duplicate or representative, we compare its witnessing points p_a and p_b against witnessing points p'_a and p'_b of every known representative pair. If the distance between p_a and p'_a and the distance between p_b and p'_b are both greater than the tolerance κ_c, the pair is declared as a new representative pair.

5.2 Localized Collision Detection

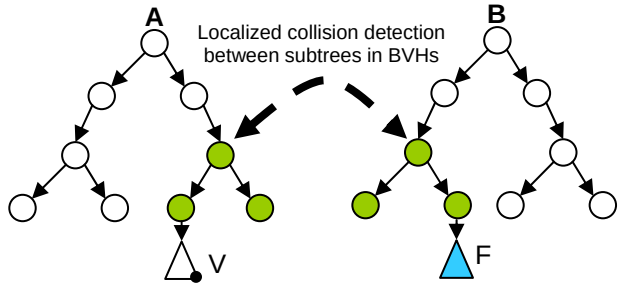


Fig. 8: Localized Collision Detection: Our algorithm can quickly cull in-colliding samples generated near the contact space and in narrow passages. The culling is achieved by checking the collision among subtrees of BVHs. We determine such subtrees using the closest features from the latest contact query (a (V,F) feature pair in this example).

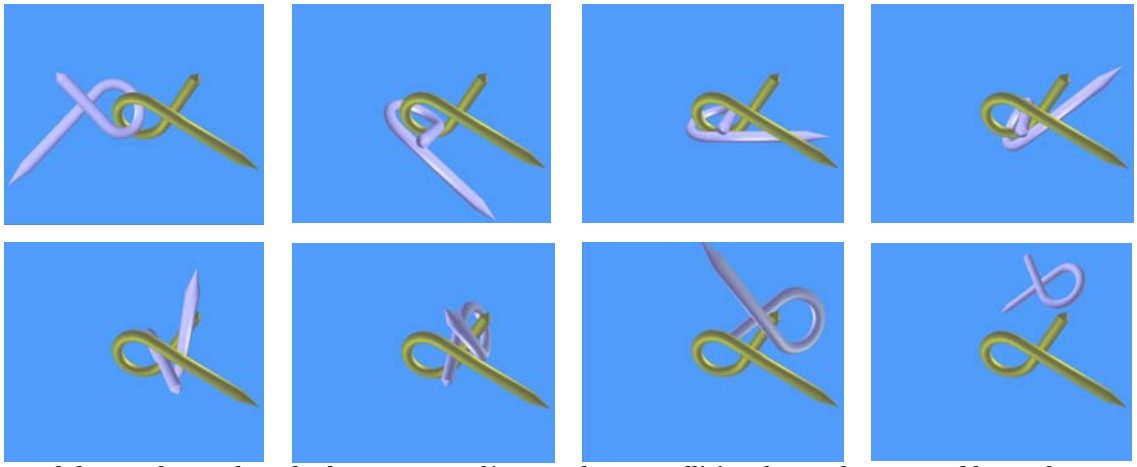


Fig. 9: Alpha Puzzle Benchmark: the sequence of images shows a collision-free path computed by D-Plan. For this challenging benchmark, D-Plan only takes 1, 043s.

The main issue with our retraction-based sampling method is how to efficiently search over the local contact space. Our method randomly generates many samples on the local contact space and performs the optimization step. In this case, the collision detection routine is invoked for every generated sample to check whether it is in free space. Given a highly cluttered environment with narrow passages, typically many generated samples would be in-colliding. When a sample is likely to lie in the C-obstacle space rather than free space, BVH-based collision detection algorithms often perform poorly as they traverse the hierarchy all the way to the leaf nodes. As a result, the resulting sampling method spends a significant fraction of the overall running in collision checking and traversing the BVHs.

We use a simple localized approach to accelerate the sample generation and collision checking. Our algorithm can conservatively cull away the samples lying in C-obstacle space by efficiently exploiting spatial coherence between nearby queries. During each retraction step, our localized collision detection algorithm performs as follows (Fig. 8):

1. Use the feature pairs reported by the latest contact query, and locate their corresponding triangles as well as the corresponding bounding volumes (BVs) in BVHs of the robot A and the obstacles B;
2. Compute the subtree within each BVH, which contains the located BVs;
3. Perform collision detection among the subtrees of BVHs for A and B;
4. If a collision is reported, we can quickly declare that the sample is in-colliding;
5. Otherwise, detect collisions using the entire BVHs.

Our localized approach can considerably accelerate the collision detection when sampling near the contact space and in narrow passages. The size of each subtree is much smaller than the entire BVH. However, our localized approach is conservative. If a sample could not be identified as in-colliding in Step 3, the algorithm moves to Step 5, and performs collision detection by traversing from the root of the BVH from scratch.

Essentially, our localized collision detection approach exploits spatial coherence that exists within our retraction-based sampling algorithm. When planning near the contact space or in narrow passages, our sampling algorithm performs collision queries for nearby samples in the configuration space. Our localized approach exploits this spatial coherence by making use of the subtrees of BVHs for direct traversal. Therefore, our approach is able to quickly cull away many in-colliding samples. Finally, our localized approach can also improve the performance of local planning methods, where collision detection is performed on a finite number of samples on the interpolated path.

6 Experimental Results

In this section, we address some implementation issues, and highlight the performance of our algorithm on different benchmarks.

Benchmarks	*Alpha Puzzle*	*Pipe*	*Wiper*	*Car Seat*
A	Alpha	Pipe	Wiper	Seat
B	Alpha	Machinery Room	Windscreen	Car Body
# Tri: A	1,044	10,352	15,197	30,790
# Tri: B	1,044	38,146	11,569	214,337

Tab. 1: Model Complexity of Our Benchmarks: Many benchmarks we use have no connectivity information. The moving objects correspond to robots (row A) and the static obstacles are shown in row B.

	Benchmarks	*Alpha Puzzle*	*Pipe*	*Wiper*	*Car Seat*
D-Plan	timing (s)	1,043.0	124.7	1197.8	181.2
	# samples	53,535	2,539	8,890	1,352
Basic RRT	timing (s)	>119,668.4	1,444.0	>12,011.3	311.0
	# samples	>84,847	4,345	>39,962	3,230

Tab. 2: Performance of D-Plan for Part Disassembly Applications: For all these difficult benchmarks, our planner is able to compute a collision-free motion in less than 20 minutes. The basic RRT planner, however, can not solve the alpha puzzle benchmark within 100 times of our planning running time. For the rest of benchmarks, D-Plan is also significantly faster than the basic RRT planner.

6.1 Implementation

Overall, D-Plan is based on RRT-based motion planning algorithm and consists of two new components: a retraction-based sample generation and a local planner enhanced by constrained motion interpolation. We implement the contact query on general polygon soup models that is needed for both the components. This enables our D-Plan to automatically handle any type of CAD models. By extending PQP [22], our contact query uses hierarchies of swept sphere volumes of given models to efficiently determine those feature pairs (V,F), (F,V), or (E/E), whose distances are less than the tolerance κ_c. In our experiment, this tolerance is simply set as the radius of the smallest enclosing sphere of the robot multiplying by 0.01. We remove the duplicate pairs and further identify locally closest pairs, using the tolerance $\kappa_r=10\times\kappa_c$. Though both parameters are chosen heuristically, they work well on our benchmarks. Fig. 7 depicts the result of contact query between two alpha-shaped models.

We use localized collision detection to improve the performance of D-Plan. Based on PQP, we find the leaf nodes in BVHs, which correspond to the latest contact query. We determine the subtrees containing the leaf nodes in each BVH. Currently, the depth of each subtree is chosen as 6, though it may be worthwhile choosing the depth according to the complexity of the model. We use the subtrees to localize the computation of collision checking.

6.2 Benchmarks

We have applied D-Plan to four challenging benchmarks. All the timing was taken on a PC with a 4-core Xeon 3GHz CPU with 4G RAM. The geometric complexity of them is summarized in Tab. 1.

1. **Alpha Puzzle** - Fig. 9: A well known benchmark with narrow passages for testing the performance of motion planning approaches [2];
2. **Pipe** - Fig. 10: Maintainability test in the CAD model of a complex machinery room. We check how to remove a pipe-shaped robot from the machinery room without any collision;
3. **Wiper** - Fig. 11: Maintainability test of the windscreen wiper motion [9]. This is an industrial benchmark with narrow passages;
4. **Car Seat** - Fig. 12: Disassembly of a seat outside a car body [9]. This is an industrial benchmark with complex geometric representation.

For simplicity, in our local planning algorithm, we perform collision detection by using a finite number of samples on the interpolating motion, e.g. 15, 50, 20, and 30 samples for alpha puzzle, pipe, wiper, and car seat benchmark, respectively.

The performance of our approach is summarized in Tab. 2. For all these difficult benchmarks, our planner takes less than 20 minutes to compute a collision-free path. Furthermore, for the alpha puzzle benchmark, D-Plan can find a collision-free path within 1,043.0 seconds, while the recent retraction-based planner takes 5,850 seconds on a relative slower machine [5], and the basic RRT planner can not find a path within 119,668.4 seconds. For the rest of benchmarks, D-Plan is also significantly faster.

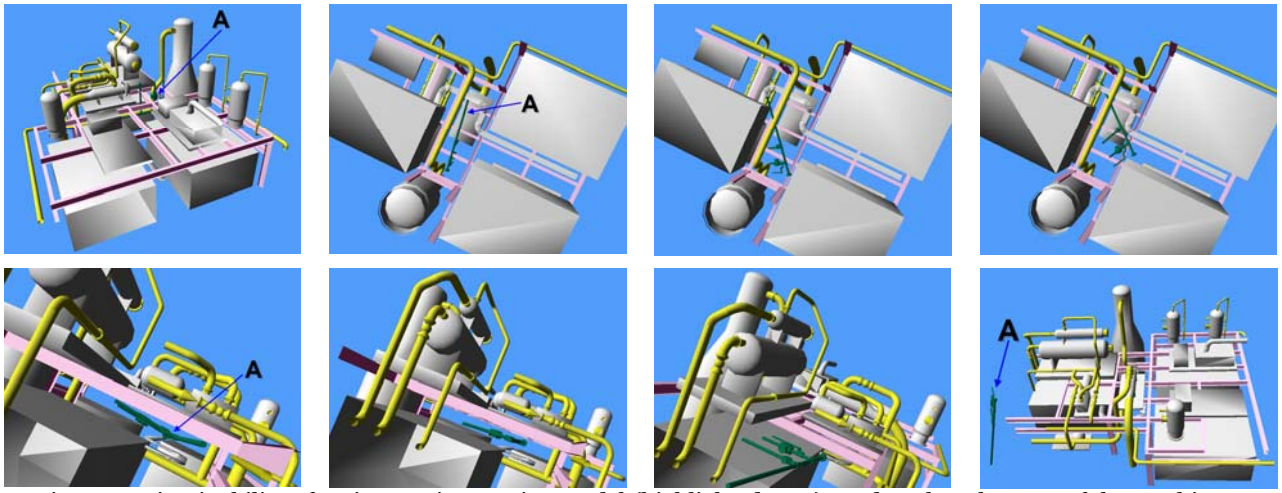


Fig. 10: Maintainability of a Pipe Motion: A pipe model (highlighted as A) needs to be taken out of the machinery room. The sequence of images shows a path automatically generated by D-Plan. It only takes around 2 minutes to compute the collision-free path.

		Alpha Puzzle	*Pipe*	*Wiper*	*Car Seat*
Contact Query	t_{con}(s)	128.2	7.1	344.8	21.4
	$\#_{con}$	82,309	4,628	21,583	1,597
	t_{per_con}(ms)	1.6	1.5	16.0	13.4
	t_{con}/t_{all}	12.3%	5.6%	28.8%	11.8%
Localized Collision Detection	t_{lcd}(s)	181.6	4.7	33.2	2.4
	$\#_{lcd}$	5,854,345	304,774	1,445,944	86,785
	t_{per_lcd} (ms)	0.031	0.016	0.063	0.028
	t_{lcd}/t_{all}	17.4%	3.7%	2.8%	1.3%
	Culling Ratio	48.8%	18.8%	36.5%	32.6%
	t_{per_lcd}/t_{per_cd}	13.3%	3.6%	9.5%	1.3%
Collision Detection	t_{cd}(s)	698.1	111.2	608.9	126.0
	$\#_{cd}$	2,995,802	247,419	917,818	58,497
	t_{per_cd}(ms)	0.233	0.449	0.663	2.1
	t_{cd}/t_{all}	66.9%	88.3%	50.1%	69.5
D-Plan	t_{all}(s)	1043.0	125.9	1197.8	181.2

Tab. 3: The table highlights the timing breakdown for D-Plan. t_{con}, $\#_{con}$ and t_{per_con} are the total timing, the number, and the timing on average for the contact query; t_{lcd} and t_{cd} are the total timing for localized collision detection and collision detection, respectively. Overall, the module for collision detection takes around 50.1% to 88.3% of the total timing (t_{cd}/t_{all}), the module for contact query takes around 5.6% to 28.8%, and the module for localized collision detection accounts for 1.3% to 17.4% of total running time. In order to evaluate the effectiveness of our localized collision detection, we measure the culling ratio $\#_{lcd}/\#_{cd}$, the number of global collision detection queries over the number of the localized collision detection queries. According to the table, our method can achieve 18.8% to 48.8% culling ratios on the tested benchmarks. Furthermore, t_{per_lcd}/t_{per_cd}, the ratio of the timing for localized collision detection over the average time for collision detection is around 1.3% to 13.3%. These two ratios indicate that our localized collision detection algorithm effectively exploits the spatial coherence and considerably improves the planner's performance.

6.3 Limitations

Our approach has a few limitations. Like most previous motion planning approaches, our local planning algorithm performs discrete collision detection along a finite number of samples on the interpolating motion. If the chosen resolution is not high enough, a local path reported as collision-free may not be correct. Furthermore, our method needs to perform the contact query repeatedly and use it for sample generation as part of the retraction step. This may impact the overall performance of the planner on complex models. Finally, D-Plan is only able to handle rigid objects.

7 Conclusions and Future Work

We present a general and efficient motion planning approach, D-Plan, for part removal and disassembly simulation. Our algorithm is based on sample-based motion planning and we use new optimization-based technique to generate samples near the boundary of C-obstacle space and narrow passages. Furthermore, we utilize a constrained interpolation scheme for local planning that connects nearby samples with a higher probability of computing collision-free paths. We also present efficient techniques for performing contact query among general polygon soup models and accelerate the performance based on localized collision detection. D-Plan is able to handle all CAD models with no assumptions on their connectivity or topology. We demonstrate its application to many challenging CAD scenarios.

There are many avenues for future work. We would like to improve the quality of the paths generated, in terms of smoothness or other constraints. We are interested in further improving the performance of the planner, e.g. using lazy collision detection technique. It may be worthwhile to test the performance on more complex benchmarks and integrate our approach into commercial CAD and virtual prototyping systems.

Acknowledgements. This research was supported in part by ARO Contracts DAAD19-02-1-0390 and W911NF-04-1-0088, NSF awards 0400134, 0429583 and 0404088, DARPA/RDECOM Contract N61339-04-C-0043 and Intel. Young J. Kim was supported in part by the Korea Research Foundation Grant funded by the Korean Government (KRF-2007-331-D00400) and the IT R&D program of MKE/IITA (2008-F-033-01, Development of Real-time Physics Simulation Engine for e-Entertainment). We thank E. Ferre from Kineo CAM and J.P. Laumond from LAAS-CNRS for providing benchmarks in Figs. 11 and 12.

8. REFERENCES

[1] M. J. Abrantes and S. D. Hill. Computer-aided planning of mechanical assembly sequences. Technical Report 95-07, 1996.

[2] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo. Obprm: An obstaclebased prm for 3d workspaces. Proceedings of WAFR, pages 197–204, 1998.

[3] N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. IEEE Trans. Robot. and Autom., 16(4):442–447, Aug 2000.

[4] H. Chang and T. Li. Assembly maintainability study with motion planning. In Proceedings of International Conference on Robotics and Automation, 1995.

[5] H.-L. Cheng, D. Hsu, J.-C. Latombe, and G. S´anchez-Ante. Multi-level free-space dilation for sampling narrow passages in PRM planning. In Proc. IEEE Int. Conf. on Robotics & Automation, pages 1255–1260, 2006.

[6] G. Dakin and R. Popplestone. Contact space analysis for narrow-clearance assemblies. In Proc. IEEE Symp. Intell. Control, 1993.

[7] B. R. Donald. A search algorithm for motion planning with six degrees of freedom. Artif. Intell., 31(3):295–353, 1987.

[8] E. Ferr and J.-P. Laumond. An iterative diffusion algorithm for part disassembly. In Proc. IEEE International Conference on Robotics and Automation, page 31493154, April 2004. 14

[9] E. Ferr, J.-P. Laumond, G. Arechavaleta, and C. Estevs. Progresses in assembly path planning. In PLM - Product Lifecycle Management, Jul 2005.

[10] M. Foskey, M. Garber, M. Lin, and D. Manocha. A voronoi-based hybrid planner. Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2001.

[11] M. Garber and M. Lin. Constraint-based motion planning for virtual prototyping. In Proc. ACM Symposium on Solid Model and Applications, 2002.

[12] R. J. Geraerts. Sample-based Motion Planning: Analysis and Path Quality. PhD thesis, Utrecht University, 2006.

[13] H. Hirukawa, Y. Papegay, and H. Tsukune. A motion planning algorithm of polyhedra in contact for mechanicalassembly. In 20th International Conference on Industrial Electronics, Control and Instrumentation, volume 2, pages 924–929, 1994.

[14] D. Hsu, L. Kavraki, J. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In Robotics: The Algorithmic Perspective—Proc. Workshop on the Algorithmic Foundations of Robotics (WAFR), pages 141–154, 1998.

[15] D. Hsu, J. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. Int. J. Robotics Research, 25(7):627–643, 2006.

[16] P. Isto. Constructing probabilitic roadmaps with powerful local planning and path optimization. In Proc. of IROS, pages 2323–2328, 2002.

[17] X. Ji and J. Xiao. Planning motion compliant to complex contact states. International Journal of Robotics Research, 20(6):446–465, 2001.
[18] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans. Robot. Automat., pages 12(4):566–580, 1996.
[19] J. Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. In IEEE Int'l Conf. on Robotics and Automation, 2004.
[20] J. Kuffner and S. LaValle. RRT-connect: An efficient approach to single-query path planning. In Proc. IEEE International Conference on Robotics and Automation, 2000.
[21] A. J. D. Lambert. Disassembly sequencing: a survey. International Journal of Production Research, 41:3721–3759(39), 10 November 2003.
[22] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. Technical Report TR99-018, Department of Computer Science, University of North Carolina, 1999.
[23] J. Latombe. Robot Motion Planning. Kluwer Academic Publishers, 1991.
[24] J. Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. International Journal of Robotics Research, pages 1119–1128, 1999.
[25] C. Pisula, K. Hoff, M. Lin, and D. Manocha. Randomized path planning for a rigid body based on hardware accelerated voronoi sampling. In Proc. of 4th International Workshop on Algorithmic Foundations of Robotics, 2000.
[26] S. Redon and M. Lin. A fast method for local penetration depth computation. Journal of Graphics Tools, 11(2):37–50, 2006. 15
[27] S. Rodriguez, X. Tang, J. Lien, and N. Amato. An obstacle-based rapidly exploring random tree. In Proc. IEEE International Conference on Robotics and Automation, pages 895–900, 2006.
[28] L. Sacks and R. H. Taylor. Interference-free insertion of a solid body into a cavity: An algorithm and a medical application. International Journal of Robotics Research, 15(3):211–219, 1996.
[29] M. Saha, J. Latombe, Y. Chang, Lin, and F. Prinz. Finding narrow passages withprobabilistic roadmaps: the small step retraction method. Intelligent Robots and Systems, 19(3):301–319, Dec 2005.
[30] R. Tesic and P. Banerjee. Motion modeling concepts in virtual manufacturing simulator. Journal of Advanced Manufacturing, 1:37–49, 2002.
[31] S. A. Wilmarth, N. M. Amato, and P. F. Stiller. Motion planning for a rigid body using random networks on the medial axis of the free space. In Symposium on Computational Geometry, pages 173–180, 1999.
[32] R. H. Wilson and J.-C. Latombe. Geometric reasoning about mechanical assembly. Artif. Intell., 71:371–396, 1994.
[33] J. Xiao and X. Ji. On automatic generation of high-level contact state space. International Journal of Robotics Research, 20(7):584–606, July 2001.
[34] L. Zhang and D. Manocha. Motion interpolation with distance constraints. Technical Report 08-001, Department of Computer Science, University of North Carolina at Chapel Hill, 2008.
[35] L. Zhang and D. Manocha. A retraction-based RRT planner. In IEEE International Conference on Robotics and Automation (ICRA), 2008.

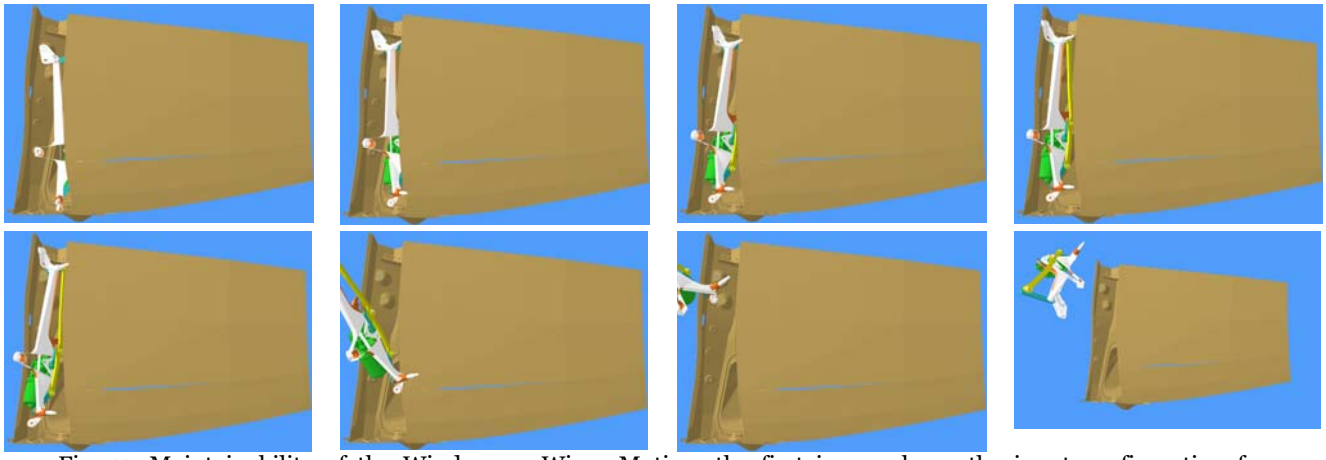


Fig. 11: Maintainability of the Windscreen Wiper Motion: the first image shows the input configuration for benchmark arising in an industrial application [8], where the wiper is inside the windscreen model with very tight space. The wiper needs to be taken outside with the final configuration shown in the last image. The intermediate sequence of images shows a path automatically computed by D-Plan.

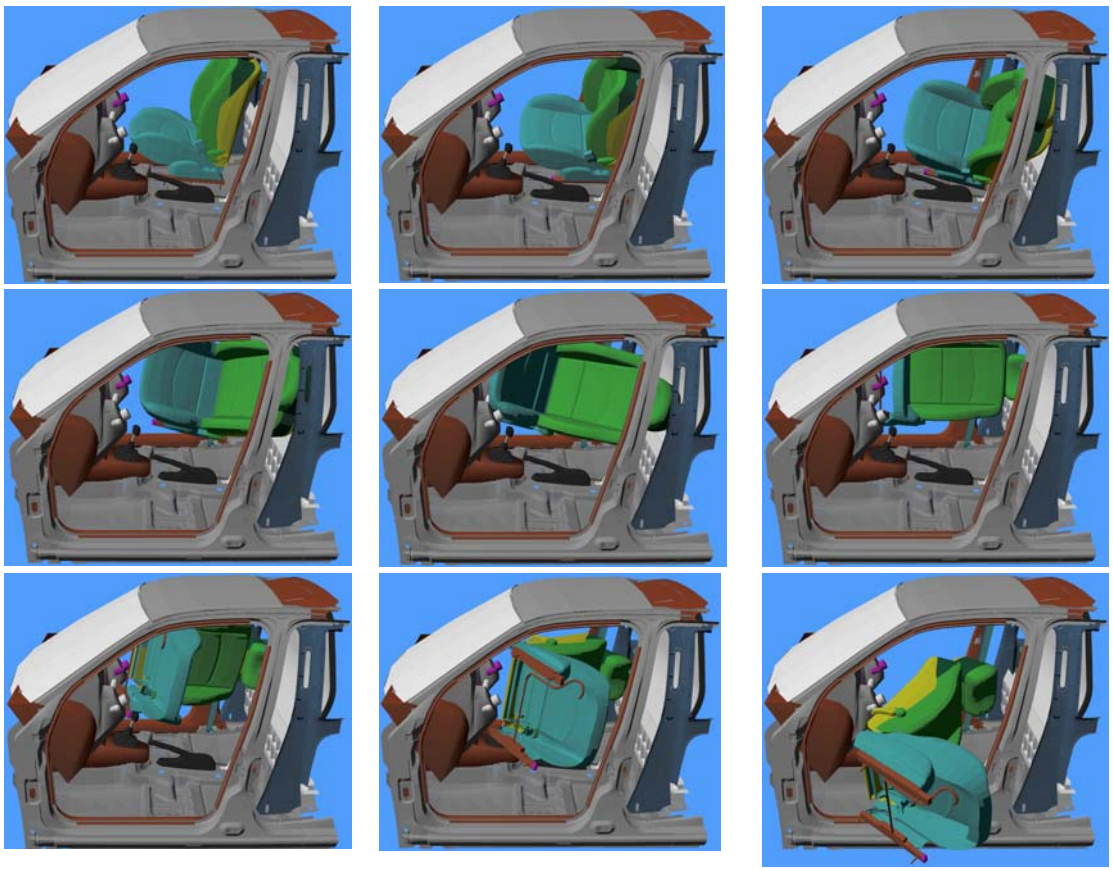


Fig. 12: Disassembly of a Seat outside a Car Body: This benchmark of disassembly of a seat outside of a car body also arises from an industrial application [8]. It is a difficult scenario for motion planning approaches due to the cluttered environment and complex geometric representation of the models, i.e. 30K triangles for the seat model and 214K triangles for the car body model. The sequence of images shows a path automatically generated by D-Plan in about 3 minutes.