Disassembly Sequencing Using a Motion Planning Approach*

Sujay Sundaram
Dept. Mechanical Engineering
sps7711@cs.tamu.edu

Ian Remmler
Dept. Computer Science
irr5509@cs.tamu.edu

Nancy M. Amato
Dept. Computer Science
amato@cs.tamu.edu

Technical Report TR00-027
Department of Computer Science
Texas A&M University
College Station, Texas, 77843-3112
October 26, 2000

Abstract

We propose a new approach for generating disassembly sequences. Our motion planning based approach treats the parts in the assembly as robots and operates in the composite configuration space of the parts' individual configuration spaces. Randomized techniques inspired by recent methods proposed for motion planning are used to sample configurations in this space. Although a purely randomized approach to sampling is successful in systems with a small number of parts, typical assemblies consist of numerous parts and the corresponding composite C-spaces have high dimensionality. In such cases, a completely randomized sampling approach would be ineffective since many important configurations for the disassembly sequence will involve closely packed parts, i.e., the disassembly sequence will pass through narrow passages in the C-space. Our solution to this problem is to bias the sampling by computing potential movement directions based on the geometric characteristics of configurations known to be reachable from the assembled configuration (the start). For example, we select potential directions which are perpendicular to part faces. Thus, our disassembly algorithm constructs a disassembly tree which is rooted at the starting assembled configuration. Our experimental results with several non-trivial puzzle-like assemblies show the potential of this approach.

*This research supported in part by NSF CAREER Award CCR-9624315 (with REU Supplement), NSF Grants IIS-9619850 (with REU Supplement), ACI-9872126, EIA-9975018, EIA-9805823, and EIA-9810937, and by the Texas Higher Education Coordinating Board under grant ARP-036327-017.

1 Introduction

Disassembly planning is the problem of finding and sequencing the motions that can separate the individual parts of an assembly. Disassembly planning has a number of applications, for example, in product maintenance and product recycling, and in particular, assembly planning. A lot of assembly planners use the assembly-by-disassembly strategy [4, 13]. An assembled product exhibits more constraints on its components than in its disassembled state and hence reduces the range of motions that a planner must consider [26].

Assembly/Disassembly planning had traditionally been of interest to AI researchers. The geometric approach to assembly planning originated in robotics [26]. In this work, the applicability of motion planning techniques to the disassembly problem is considered.

While complete motion planning algorithms do exist, they are rarely used in practice since they are computationally infeasible in all but the simplest cases [20]. For this reason, attention has focused on randomized methods, which sacrifice completeness in favor of computational feasibility and applicability. Indeed, several very successful randomized motion planning algorithms have recently been used to solve many previously unsolved problems in high-dimensional configuration spaces, e.g., *probabilistic roadmap* (PRM) methods [1, 18], the Ariadne's Clew algorithm [5], and the Rapidly-exploring Random Trees method [22].

Disassembly seems to offer the same challenges. In particular, it can be posed as a motion planning problem in the composite configuration space composed of the configuration spaces of the individual parts [20]. As with motion planning, the high-dimensionality of this space warrants randomized methods. However, a completely randomized approach would be ineffective since many important configurations for the disassembly sequence will involve closely packed parts, i.e., the disassembly sequence will pass through narrow passages in the C-space [15].

Our approach to the disassembly problem involves biasing the sampling based on the geometric characteristics of configurations known to be reachable from the assembled configuration (the start). In particular, information based on the relative positions of the parts in one configuration is used to generate potential movement/removal directions for one or more parts. Geometric information is also used to determine which parts should or should not be moved at any given point in the disassembly process. The resulting disassembly algorithm iteratively grows a tree of configurations connected to the start (assembled) configuration, similar in nature to the Ariadne's Clew algorithm [5] and the Rapidly-exploring Random Trees method [19]. Our results on several puzzle assemblies show the promise of this new approach to disassembly.

2 Related Work

Assembly Planning, though originally considered an AI topic, was given a geometric facet by the robotics community. The assembly-by-disassembly strategy has been a common approach to assembly planning [4, 13]. Automatic Planners were developed using different schemes amongst which is the potential field approach of Gottschlich and Kak [9]. Wilson and Latombe [26] suggested a non-directional blocking graph (NDBG) for the efficient generation of assembly algorithms. Wolter [27] analyses and reports different schemes and data structures used in assembly planning. Constraint Languages have also been commonly used for assembly planning [28, 16]. Disassembly planning has applications particularly in the area of de-manufacturing and recycling. Virtual Reality has also been utilized to help realize better disassemblability [7].

Halperin, Latombe and Wilson [10] used a motion planning approach with the NDBG to realize a motion space scheme, which is similar to the configuration space scheme used in motion planning.

3 Composite C-Spaces

Composite configuration spaces are a convenient and direct way to extend the configuration space concept to multiple robots [20]. The m individual robots $A_1, \ldots, A_m$ are treated as a single multi-bodied (composite) robot $A = \{A_1, \ldots, A_m\}$. The configuration space of A is

$$\mathcal{C} = \mathcal{C}_1 \times \mathcal{C}_2 \times \ldots \times \mathcal{C}_m$$

where $\mathcal{C}_i$ is the configuration space of the individual robot A_i, for $i = 1 \ldots m$. A configuration $q \in \mathcal{C}$ corresponds to a unique position and orientation for *each* robot A_i. That is, $q = (q_1, \ldots, q_m)$, where $q_i \in \mathcal{C}_i$. C-space obstacles (C-obstacles) result from collisions between a robot and an obstacle, and also collisions between robots.

Composite configuration spaces are a natural representation for assemblies. In particular, each part in the assembly is considered to be a robot, each with its own configuration space. There may or may not be other obstacles in the environment.

Motion planning techniques that operate in C-space can often be naturally extended to composite C-spaces. A complication, however, is that the composite C-spaces can have very high-dimensionality (the dof is the sum of the dof of the individual C-spaces). For example, if an assembly consists of m rigid parts, then the C-space of each part would have 6 dof, and the composite configuration space of all parts in the assembly would have $6m$ dof. Thus, care must be taken in determining motion planning methods appropriate for composite C-spaces.

While we find it convenient to work in the composite configuration space, which contains all the degrees of freedom, we will take advantage of the knowledge that $\mathcal{C}$ is composed from the $\mathcal{C}_i$ of the individual robots. In particular, we will use this knowledge to selectively sample the C-space. For example, by 'masking' off all dof except those belonging to a particular robot(s), we can control which robots move and which do not. As we will see, this will be extremely useful in the disassembly problem where the parts are packed together in highly constrained configurations, with some parts blocking the movement of other parts. In this case, one could mask off the parts in the blocked configurations from the sampling until other parts have been moved away.

4 The Naive Approach

Given our composite configuration space formulation of the assembly, it is natural to ask if known motion planning methods can be applied to solve the disassembly problem. While this approach had been previously considered, it was not very successful due to the lack of practical motion planning methods for high-dimensional C-spaces. However, there have been some recent advances in this regard. In particular, the *probabilistic roadmap (*PRM*)* motion planning methods [18] have proven to be very successful in exploring high-dimensional configuration spaces (see, e.g., [2, 3, 11, 17, 24]). The idea behind these methods is to create a graph (roadmap) of randomly generated collision-free configurations which are connected by a simple and fast local planning method. Queries are answered by connecting the start and the goal to the roadmap, and finding a path in the roadmap connecting them. The PRM approach is also appealing since it is simple and easy to implement, requiring only collision detection as a primitive operation.

Based on these successes, our first approach to the disassembly problem is a naive application of the PRM paradigm. In the roadmap construction phase (preprocessing), we randomly sample configurations from $\mathcal{C}$, retain those that are collision-free, and connect them using a simple local planner (e.g., one that connects along a straight-line in C-space). In the query phase, we attempt to connect the start (the assembled configuration) and the goal (a disassembled configuration) to the same connected component of the roadmap, and then extract a path between them.

I. Roadmap Construction Phase

A. Node Generation phase

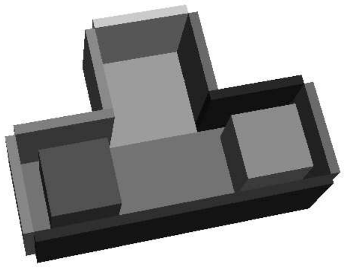
1. `for` $i = 1$ `to` n `(number of nodes to generate)`
2. `generate a` cfg `at random in C-space`
3. `if (`cfg `is collision-free)`
4. `add` cfg `to` V `(roadmap vertices)`

B. Node Connection Phase

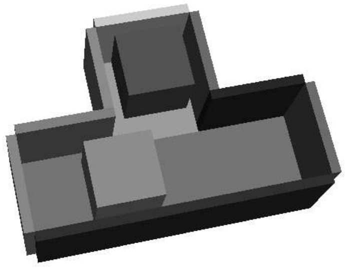
5. `for` $i = 1$ `to` $|V|$
6. $N_i = k$ `closest cfgs to` v_i `in` V
7. `for each vertex` $n \in N_i$
8. `if (local planner connects` v_i `to` n`)`
9. `add` (v_i, n) `to` E `(roadmap edges)`

II. Query Phase

1. `Try to connect start and goal to roadmap` R
2. `search for a path between connection points`
3. `if (path exists)`
4. `return path`
5. `else return failure`



(a)



(b)

Figure 1: (a) A cube swapping problem, and (b) a snapshot of the solution path.

A typical problem that was solved using this method is shown in Figure 1. The objective of this puzzle is to swap the positions of the two cubes. The composite C-space has 12 dof, six for each cube. In this naive approach, random configurations in $\mathcal{C}$ are generated (line 2) by generating a random value for each of the 12 dof. Note that this implies that all edges (and hence paths) in the roadmap will simultaneously move both cubes.

Recall that as the number of parts increases, the dimensionality of the configuration space increases as well. We can also observe that for the disassembly problem, many important configurations in the disassembly sequence will have multiple parts packed together in very constrained situations, e.g., the starting assembled configuration. Such configurations are contained in what are called *narrow passages* in C-space [14], and are known to be difficult to generate using PRM methods. Due to these challenges, it is unlikely that a method based solely on random sampling will prove effective for the disassembly problem. In the next section, we discuss techniques that can be used to guide the sampling to achieve better results.

5 The Disassembly Algorithm

A typical disassembly would consist of numerous parts and the corresponding composite C-space would have high dimensionality. As discussed above, a completely randomized sampling approach would be ineffective since many important configurations for the disassembly sequence will involve closely packed parts, i.e., the disassembly sequence will pass through narrow passages in the C-space [14]. In the past, randomized techniques have been combined with more deliberate methods to great success. For example, in [11], the PRM methodology was used in conjunction with forward and inverse kinematics for problems involving closed chain systems, [12] uses energy minimization techniques in a PRM–based approach for flexible objects, and biased sampling around the (known) goal configuration was used to compute protein folding pathways in [25].

Our approach to the disassembly problem involves biasing the sampling based on the geometric characteristics of configurations known to be reachable from the assembled configuration (the start). In particular, information based on the relative positions of the parts in one configuration will be used to generate potential movement/removal directions for one or more parts. We will also use geometric information to determine which parts should or should not be moved at any given point in the disassembly process.

A high level description of our disassembly algorithm is given below. The roadmap graph $R = (V, E)$ consists of vertices V (collision–free configurations) and edges E (valid paths between two configurations). S is a stack of configurations, initially containing only the assembled configuration, that are recursively 'expanded' until a disassembly is obtained. The choice of a stack indicates our preference for a 'depth-first-search' exploration from the starting assembled configuration.

DISASSEMBLE(CFG c_0)

```
S = initial configuration c_0
R = {(V = c_0), (E = ∅)}
while ( (|S| ≠ 0) and !IsDisassembled() )
 c = popped element from S
 NewCfgs = expand(c)
 for each c_i ∈ NewCfgs
    push back c_i to S
    add c_i to V
    add edge(c, c_i) to E
```

The `IsDisassembly()` function determines if the system of parts is in a disassembled state. Disassembly is assumed if each part is beyond a pre-defined distance from all others. The `expand(c)` method, described below, generates configurations that progressively disassemble based on a configuration c. The key challenge here is determining promising directions in which to expand.

EXPAND(CFG c)

```
Determine a set of directions in C-space
for each direction d
```

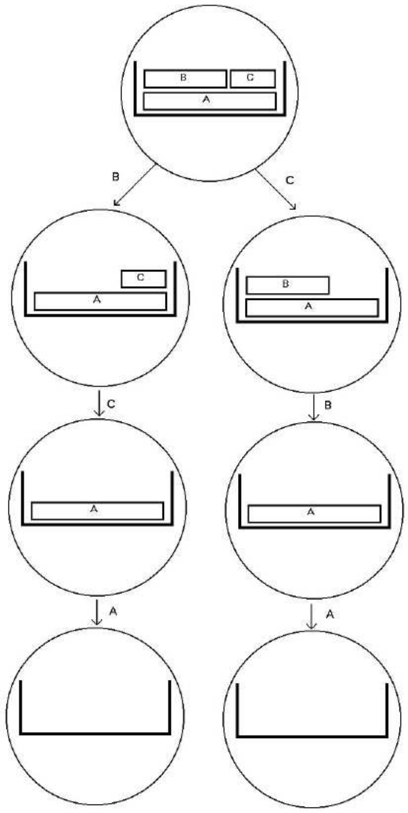


Figure 2: A disassembly roadmap for a simple problem.

```
  move incrementally from c in dir d
  if (there is no collision)
     add new cfg c' to GenCfgs
return GenCfgs
```

This approach, which iteratively generates a disassembly sequence, differs significantly from the naive PRM approach. Although it too builds a roadmap, the roadmap's growth and structure are different from a PRM roadmap. In particular, this approach grows a connected component of configurations (actually a tree) from the starting assembled configuration (see Figure 2). Conceptually, the node generation, node connection, and query phases are combined into a single stage. The generation of new configurations by 'expanding' configurations already in the roadmap means no separate connection stage is needed since each configuration is connected to the one it was expanded from, and by transitivity, to the starting assembled configuration. This is similar in spirit to randomized methods like the Ariadne's Clew algorithm [5] and the RRT approach [21] which also build trees of connected configurations from the start configuration.

5.1 Determining movement directions

The key to the success of the disassembly algorithm lies in the selection of potential removal directions for the configuration chosen for expansion. Note that the removal direction is in fact a direction in $\mathcal{C}$, and thus may involve movement by one or more parts simultaneously. Thus, there are in essence two decisions that must be made when selecting movement directions: which part(s) to move, and in which direction to move the selected part(s). In fact, it is sometimes useful to consider the second question when answering the first. That is, it is sometimes easier simply to decide which parts have good potential removal directions, and then try them.

For assemblies, such as the puzzles considered in this paper, which contain rectilinear parts, it can be observed that in many cases the disassembly of a part in a system can be carried out in a direction perpendicular to one of its faces. Hence, the face normals should be considered as potential removal directions. As will be discussed in Section 6, this heuristic proved its efficacy for the problems studied.

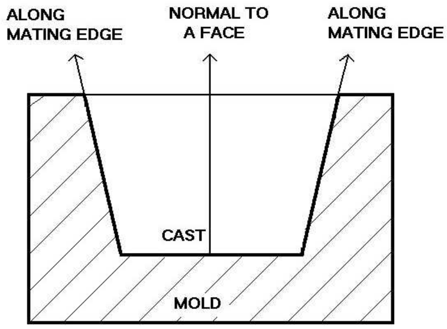


Figure 3: A 'casting' problem showing removal directions.

The use of the face normals as potential removal directions is related to the *casting problem* [6], which arises in manufacturing and has been studied in computational geometry, among other fields. The casting problem involves the removal of a cast from a mold; the generation of possible removal directions based on the geometry of the mold. This is relevant to the disassembly problem as the latter involves a system of parts that are closely meshed with each other. Algorithms for the casting problem consider directions along the mating edges/faces of the cast and mold as potential removal directions (see Figure 3). In future work, we plan to include such directions in our set of potential movement directions.

Both the above approaches implicitly assume one is moving only one part at a time. This is done by masking out the dof of the remaining parts and altering only the dof of the part to be moved. Assemblies which can be disassembled in this manner have what is called a *serial disassembly sequence*.

In some cases, however, it is necessary to move more than one part at a time. As serial disassemblies are generally preferred, our approach is to first try to generate a disassembly sequence with this property. If one cannot be obtained, then simultaneous movement of two parts is attempted. If this too fails, then simultaneous movement of three parts is attempted, and so on.

An intermediate case between movement of one part, and movement of multiple parts, is to consider moving *subassemblies*, that is, sets of parts that can be moved together. Here one must first identify promising subassemblies, and then, considering them as one merged part, generate potential removal directions for the entire subassembly, just as was performed for the individual parts.

6 Results

In this section we examine the performance of our method on several disassembly puzzles.

Model	dof	Time (s)	Nodes	Edges
Swap Cubes (naive)	12	337	359	331
Swap Cubes (intelligent)	12	10	15	14
'Planar' puzzle with plates	36	232	170	171
'Planar' puzzle with box	36	474	209	211
3D 10 block puzzle	60	2523	541	543
Pentomino puzzle	72	21794	703	705

Table 1: Execution statistics on the various models.

6.1 Implementation details

We implemented the disassembly algorithm described in Section 5, and our `expand()` subroutine uses face normals as potential removal directions.

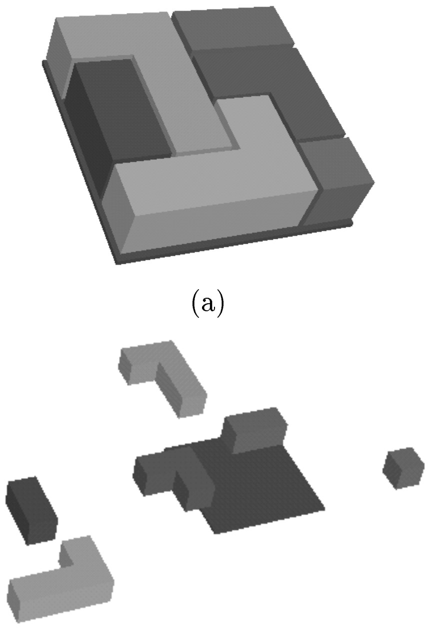
Our system was written in C++, and was built on top of the C++ OBPRM library [1] which implements our OBPRM planner and our interpretations of a variety of other PRMs, and supports several collision detection libraries such as RAPID [8], V-Clip [23], and the C-Space Toolkit (CSTK) [29, 30]. New classes were designed to handle composite configuration spaces.

6.2 Experimental Results

Comparison with the Naive Approach. The model shown in Figure 1 (top cover not shown for clarity) was tested with both the naive PRM approach and our more 'intelligent' disassembly algorithm. Although the purely randomized (naive) scheme did solve the puzzle, it was not very efficient as shown in Table 1. Also, the number of nodes that needed to be sampled to result in a positive query was high (around 5000 in this case). In contrast, our disassembly algorithm solved the problem very quickly. This example illustrates the benefits that can be obtained by providing randomized approaches with some intelligent means of directing their sampling.

'Planar' Block Puzzles. We also tested our disassembly algorithm on some other 'planar' puzzles (while they are really three-dimensional puzzles, the assembled puzzle lies on a plane). The naive approach was not applied on these due to its observed weaknesses on the cube swapping puzzle. The first is a planar-blocks puzzle shown in Figure 4(a). The motion of the blocks is restricted by a couple of plates (the top one is not shown in the figure for clarity). Figure 4(b) shows a snapshot of the output path. Note that the movement of the individual blocks in the snapshot of the solution path is in a direction normal to one of its faces. As can be seen in Table 1, this problem was solved quite easily by our algorithm.

The same planar-blocks model was also studied in a more restricted environment: it was enclosed



(a)

(b)

Figure 4: (a) A planar blocks puzzle, and (b) snapshot of the solution path.

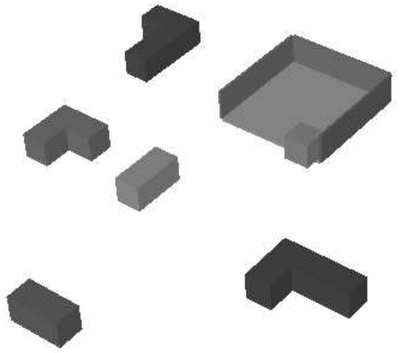
in a box with only one opening (see Figure 5, where the top plate is not shown for clarity). As expected, the increased complexity of the problem resulted in a substantial increase in computation time (see Table 1). An interesting fact to note is that since the number of possible motions of the initial configuration is limited, the resulting roadmap tree is 'narrower'.

Three-Dimensional Puzzles. Two other three-dimensional models were also tested. The first has 10 pieces (Figure 6(a)) and the other was a Pentomino puzzle (so called because each piece is made of 5 cubes) with 12 pieces (Figure 7(a)). The disassembled configurations are shown in Figure 6(b) and Figure 7(b), respectively. As can be seen in Table 1, these puzzles require significantly more time than the others tested, which is not surprising given their higher dimensionality. However, this is not the entire explanation since the 72 dof Pentomino Problem required significantly more time than the other puzzle which has 60 dof. In fact, the disassembly sequence for the Pentomino puzzle is more constrained, resulting in a C-space with more 'narrow corridors'. Our current implementation does not yet attempt to determine the best part(s) to move at any given stage, and so it suffers some performance loss for a highly constrained problem like the Pentomino puzzle.

7 Conclusion and Future Work

The disassembly algorithm presented here represents a new approach to (dis)assembly planning. While the applicability of motion planning techniques to disassembly planning had been considered previously, it did not lead to practical algorithms. This was mainly due to the lack of motion planning algorithms suitable for high-dimensional, highly constrained, configuration spaces. Our approach to the disassembly problem is inspired by recent advances in randomized motion planning

(a)



(b)

Figure 5: (a) Planar blocks puzzle in a box, and (b) path snapshot.

algorithms. We bias the sampling based on geometric characteristics of configurations known to be reachable from the assembled configuration (the start). The resulting disassembly algorithm iteratively grows a tree of configurations connected to the start (assembled) configuration.

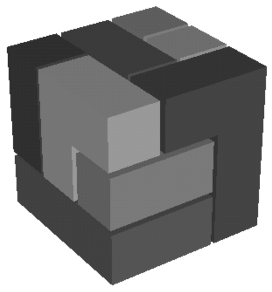
Our experimental results on several puzzle assemblies show the promise of this new approach to disassembly. In current work, we are exploring additional techniques for generating potential movement directions for parts in the assembly, and to determine which parts are good candidates for movement. In particular, we note that all the puzzles studied here were solved using face normals as potential removal directions. While these were sufficient for these puzzles, we anticipate the need for different heuristics for different types of assemblies. This is the subject of current work. In the future, we plan to investigate the application of our techniques to direct assembly planning rather than for assembly-by-disassembly.

Acknowledgements

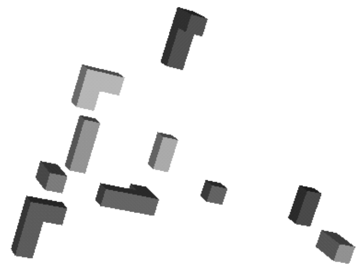
We would like to thank the robotics group at Texas A&M University. In particular, we would like to thank Guang Song with his help integrating our assembly planner with the OBPRM library.

References

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. 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.



(a)



(b)

Figure 6: (a) A three-dimensional 10 blocks puzzle, and (b) a path snapshot.

[2] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 113–120, 1996.

[3] E. Anshelevich, S. Owens, F. Lamiraux, and L. Kavraki. Deformable volumes in path planning applications. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000.

[4] E. M. Arkin, R. Connelly, and J. S. B. Mitchell. On monotone paths among obstacles, with applications to planning assemblies. In *Proc. ACM Symp. on Computational Geometry (SoCG)*, 1989.

[5] P. Bessiere, J. M. Ahuactzin, E.-G. Talbi, and E. Mazer. The ariadne's clew algorithm: Global planning with local methods. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, volume 2, pages 1373–1380, 1993.

[6] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer, Berlin, 1997.

[7] R. Gadh, H. Srinivasan, S. Nuggehalli, and R. Figueroa. Virtual disassembly: A software tool for developing product dismantling and maintenance systems. In *IEEE Reliability and Maintainability Symposium*, 1998.

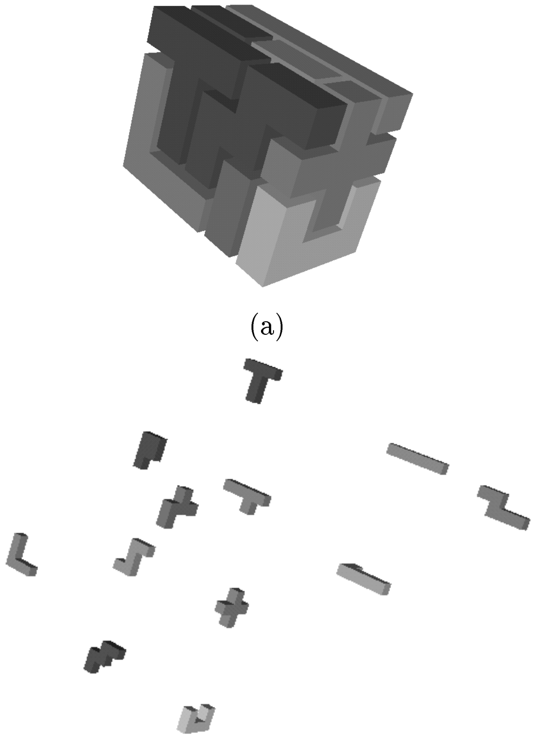
[8] S. Gottschalk, M.C. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. Technical Report TR96-013, University of N. Carolina, Chapel Hill, CA, 1996.

[9] S. N. Gottschlich and A. C. Kak. AMP-CAD: An assembly motion planning system. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1992.

[10] D. Halperin, J-C. Latombe, and R. Wilson. A general framework for assembly planning: The motion space approach. In *Proc. ACM Symp. on Computational Geometry (SoCG)*, 1998.

[11] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000.

[12] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. Technical Report TR 00-003, Department of Computer Science, Texas A&M University, 2000.



(a)

(b)

Figure 7: (a) The Pentomino puzzle, and (b) a path snapshot.

[13] R. L. Hoffmann. Automated assembly in a csg domain. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1994.

[14] D. Hsu, L. Kavraki, J-C. Latombe, and R. Motwani. Capturing the connectivity of high-dimensional geometric spaces by parallelizable random sampling techniques. In *Proc. IEEE Workshop Randomized Parallel Computing(WRPC)*, 1998.

[15] D. Hsu, L. Kavraki, J-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998.

[16] R. E. Jones, R. H. Wilson, and T. L. Calton. On constraints in assembly planning. *IEEE Trans. Robot. Automat.*, 1998.

[17] L. Kavraki, F. Lamiraux, and C. Holleman. Towards planning for elastic objects. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998.

[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.*, 12(4):566–580, August 1996.

[19] J. J. Kuffner and S. M. LaValle. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 995–1001, 2000.

[20] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[21] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.

[22] S. M. LaValle and J. J. Kuffner. Rapidly-Exploring Random Trees: Progress and Prospects. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages SA45–SA59, 2000.

[23] B. Mirtich. V-clip: Fast and robust polyhedral collision detection. Technical Report TR97-05, Mitsubishi Electric Research Lab, Cambridge, MA, 1997.

[24] A.P. Singh, J.C. Latombe, and D.L. Brutlag. A motion planning approach to flexible ligand binding. In *7th Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 252–261, 1999.

[25] G. Song and N. M. Amato. A motion planning approach to folding: From paper craft to protein folding. Technical Report TR00-017, Department of Computer Science, Texas A&M University, July 2000.

[26] R. H. Wilson and J-C. Latombe. Geometric reasoning about mechanical assembly. *Artificial Intelligence, Elsevier Science*, pages 371–396, 1994.

[27] J. Wolter. A combinatorial analysis of enumerative data structures for assembly planning. *Journal of Design and Manufacturing*, 2(2), 1992.

[28] J. Wolter, S. Chakrabarty, and J. Tsao. Mating constraint languages for assembly sequence planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1995.

[29] P. Xavier. Fast swept-volume distance for robust collision detection. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1162–1169, 1997.

[30] P. G. Xavier and R. A. LaFarge. A configuration space toolkit for automated spatial reasoning: Technical results and ldrd project final report. Technical Report SAND97-0366, Sandia National Laboratories, 1997.