The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Efficient Planning of Disassembly Sequences in Physics-Based Animation

Jacopo Aleotti and Stefano Caselli

Abstract— **We address the problem of disassembly planning from a novel perspective. In the proposed method the goal is to find all the physically admissible subassemblies in which a set of objects can be disassembled and to identify feasible disassembly motions. Stability of object configurations under the effect of gravity and friction is computed by relying on a physics-based animation engine. We propose efficient strategies to reduce computational time that take into account precedence relations, arising from user assembly demonstrations as well as geometrical clustering. We have also developed a motion planning technique for generating non-destructive disassembly paths on a query-based approach. Experiments have been performed in an interactive virtual environment including a dataglove that allows realistic object manipulation and grasping.**

I. INTRODUCTION

Assemblies are collection of objects. Disassembly planning is the problem of finding appropriate motions that are applied to the individual parts of an assembly to separate the initial agglomerate. We consider a (static) stable assembly configuration as a composition of objects that does not collapse under the effect of gravity forces and static friction. A disassembly motion of an object is feasible if the object can reach an infinite distance from its initial location without colliding with the remaining bodies and if the resulting subassembly configuration (obtained by removing the object) is still stable. A disassembly sequence is a succession of feasible motions that are applied to the objects one by one starting from an initial configuration that includes all the objects in the environment. Disassembly planning has strong applications in industrial manufacturing for recycling and for cost reduction in product dismantling. Assembly and disassembly problems are interrelated since assembly plans can be derived from disassembly sequences. Disassembly planning is a typical combinatorial problem where the total number of possible disassembly sequences is given by the factorial $n!$ of the number of objects in the initial configuration. Therefore, even a rather small environment with ten objects has a potential of more than three millions of possible disassembly sequences. Moreover, the computational complexity required to compute the stability of a given configuration is NP-hard.

In this work we propose a computational method for planning disassembly tasks which relies on physics-based animation. The method differs from the large amount of previous contributions on disassembly planning that have mainly focused on geometrical aspects. In particular, we propose an effective approach for automatic evaluation of subassembly stability and disassembly planning for models of arbitrary shape. Systematic stability evaluation is achieved by means of a physics simulation engine. The problem can be stated as follows. First, find all the physically stable subassemblies in which a collection of rigid bodies can be disassembled. Second, identify feasible disassembly sequences for all objects by applying a single-query motion planning algorithm. The adopted solution ensures realistic non-destructive disassembly motions, meaning that the objects are disassembled by looking only at continuous paths. Stability of object configurations is computed by taking into account both gravity and friction. We assume that only pure contact forces act on the bodies in the physics-based environment. We do not consider external interconnections such as forces due to connectors like glue, screws or bolts.

In order to reduce the computational time required to compute all the admissible subassemblies of a given set of objects we propose efficient strategies for identifying as many stable configurations as possible without the need of a physical simulation. Such optimizations take advantage of precedence relations arising from user assembly demonstrations. Indeed, we apply the Programming by Demonstration paradigm [9], [17], [23], [2] for collecting multiple demonstrations of the same assembly structure and we exploit redundancy to detect precedence relations. A precedence relation between the two objects means that the first object must be assembled before the second one.

Theoretical overviews of assembly precedence graphs can be found in the work of Chen and Henrioud [3] that presented a method for systematic generation of all feasible assembly precedence graphs for mechanical products. A further optimization that we propose consists of a spatial clustering algorithm that groups geometrically adjacent objects, thus allowing stability evaluation on sets of lower dimensions, and also enabling parallel processing. Experiments have been performed in an interactive virtual environment that is also exploited for visualizing the results of the automatic disassembly operations. The virtual environment also enables realistic grasping of objects by means of a simulated anthropomorphic hand driven by a dataglove.

The paper is organized as follows. Section II reviews the state of the art concerning disassembly planning. Section III describes the proposed approach and its optimizations, while section IV reports experiments. Section V focuses on the non-destructive disassembly planner. Section VI closes the paper summarizing the work.

J. Aleotti is with Dipartimento di Ingegneria dell'Informazione, University of Parma, Italy aleotti@ce.unipr.it

S. Caselli is with Dipartimento di Ingegneria dell'Informazione, University of Parma, Italy caselli@ce.unipr.it

II. RELATED WORK

De Mello and Sanderson [8] defined the AND/OR graph representation for assembly plans. Such representation constitutes the basis for several planning systems. A remarkable number of works have investigated disassembly sequence planning without considering physical issues. Mattikalli et al. [15] proposed a geometrical approach to automatically determine disassembly sequences. Waarts et al. [22] presented a sequence planner which takes into account both feasibility and accessibility of operations but does not include analysis of stability under gravity. Ong and Wong [18] introduced a method to extract subassemblies by grouping components together based on the criteria of connectivity and interference relationships. Sundaram et al. [20] applied conventional motion planning strategies for automatic disassembly. Cortes et al. [4] described a Manhattan-like motion planner based on Rapidly-exploring Random Tree (M-RRT) suitable for disassembly of objects with articulated parts. The use of liaison graphs for generation of mechanical assembly sequences has also been frequently considered to describe connectivity between assembly parts [10], [5]. Aguinaga et al. [1] proposed the targetless RRT (T-RRT) algorithm which has been adopted as a component of our physics-based disassembly planning system (section V). In [7] a general framework for disassembly planning is presented which is based on the concept of motion space. Torres et al. [21] developed a disassembly sequence planner that exploits information encoded as precedence relations introduced by an operator who knows the product. Gadh et al. [6] proposed a virtual disassembly tool for product dismantling that involves a cost metric. The importance of taking into account stability in disassembly planning was pointed out by Lee and Yi [12] that illustrated different criteria aimed at reducing the search space by pruning unstable subassemblies. In [11] an assembly planning system has been designed that includes physical reasoning about interconnection forces. Loomis and Balkcom [13] introduced an efficient approach which is based on computational reuse of rigid-body dynamics but is limited to 2D environments. Several authors have also investigated theoretical approaches for finding all potential stable orientations of an assembly under the effect of gravity and Coulomb friction [14], [16].

III. EFFICIENT STABILITY EVALUATION

This section describes the proposed method for finding all the physically stable subassemblies of a collection of objects and its optimization. The algorithm also computes all the possible destructive disassembly sequences. A second phase, discussed in section V, performs non-destructive disassembly planning on a query-based approach for selected sequences. Stable subassemblies are organized into a disassembly graph. A disassembly graph of a set of n objects $P = \{p_1, \ldots, p_n\}$ is defined as a directed graph $G = \{X, C\}$ where X is the set of nodes corresponding to the stable subassembly configurations (partitions of P), while C represents the set of arcs. Each arc is an oriented edge that connects two configurations X_i and X_j, where the stable configuration X_j can be obtained from X_i by removing one object. Configuration X_j is called a child node of X_i. The initial state $X_0 = \{p_1, \ldots, p_n\}$ comprises all the objects in the environment. The terminal nodes are all the stable configurations that comprise only one object. A stable destructive disassembly sequence is given by any path of nodes from the root to the terminal nodes of the graph. Figure 1 shows the interactive virtual environment with one example of stable configuration as well as one unstable collapsing configuration.

Algorithm 1 reports the pseudo-code for generating the disassembly subgraph of a generic node, while Algorithm 2 reports the procedure used for testing the stability of a node. The disassembly phase starts by invoking Algorithm 1 on the root node of the graph ($compute\_disassembly\_graph(X_0)$), which contains all the objects in the environment. The disassembly graph is generated iteratively by computing the stability of each configuration. Each child node is generated by removing an object from the parent configuration. Unstable configurations are removed from the graph and their children nodes are pruned. Algorithm 2 shows that in order to evaluate the stability of a configuration the system performs a physics-based simulation of the subassembly for a predefined time and computes, at each step, the linear and angular velocities of all the bodies in the environment. If both velocities do not exceed predefined small thresholds within the simulation period the configuration is considered stable. If the velocity limits are exceeded then the simulation is stopped and the configuration is considered as unstable. Optimizations A and B are described later in this section. To improve the performance of the disassembly algorithm the graphical output is disabled in this phase.

The bottleneck of the disassembly algorithm is clearly the physics-based disassembly routine. Without any optimization the computational time required to compute all possible disassembly sequences may quickly become excessive, due to the combinatorial nature of the problem. One might be tempted to reduce the computational time required to evaluate the stability of a configuration by simply reducing the time period or the velocity thresholds. However, such constants can not be arbitrarily reduced, as this would lead to the generation of false positive stable configurations. Therefore, to speed-up the disassembly phase we introduce efficient strategies to automatically assess the stability of a configuration. The idea is to perform the physics-based simulation for as few configurations as possible. The proposed optimization methods take into account multiple user demonstrations of assembly sequences that generate precedence relations between assembled objects, as well as geometrical object clustering. The user can provide assembly demonstrations by performing grasping and placing operations in a physics-based virtual environment. Indeed, the virtual environment includes a simulated anthropomorphic hand, driven by a dataglove and motion tracker, which can grasp the objects. The optimization strategies are discussed in the following.

- Optimization A is a trivial use of prior knowledge about stable configurations. Configurations that are known to be stable in advance are subassemblies extracted from user demonstrations. Each complete user demonstration of an assembly configuration X_i that comprises n objects provides a prior knowledge of n subassembly configurations that are stable. In case optimization A is enabled, Algorithm 2 avoids the physics-based test if the current configuration belongs to the set of the configurations that are known to be stable.

- Optimization B exploits precedence relations between objects. There exists a precedence relation between two objects (o_a, o_b) if o_a (pre-condition) must be assembled before o_b (post-condition). Each complete user demonstration of an assembly configuration X_i that comprises $n \geq 2$ objects provides a total of $\sum_{k=1}^{n}(k-1) = \frac{n(n-1)}{2}$ precedence relations. The set R of all precedence relations is reduced as multiple demonstrations are provided due to generalization of task constraints. Let X_j be a configuration obtained from X_i by removing object o_r. If o_r is not a pre-condition in any precedence relation involving other objects contained in X_i then X_j is automatically stable. Formally if $\forall o_i \in X_i \setminus \{o_r\}$ $\nexists(o_r, o_i) \in R$ then X_j is stable. In other words, given an assembly configuration, the configuration obtained by removing an object that is not expected to be assembled before any other object is always stable. It is trivial to show that optimization B implies optimization A, meaning that all stable configurations that can be identified by applying optimization A can also be found by optimization B. However, optimization A is slightly faster.
- Optimization C is based on an object clustering approach. An algorithm has been developed that identifies clusters of adjacent objects by means of collision detection (another approach for object clustering is proposed in [19]). If a configuration may be split into at least two clusters then clusters are disassembled separately in sequential order. A merging procedure is then applied that automatically generates all the stable configurations from the partial subtrees without the need of a physics simulation.
- Optimization D performs clustering and parallel multi-threaded computation of the identified clusters on a multi-core CPU. Recursive clustering has been limited by empirical assumptions on the number of objects in the clusters to avoid excessive overhead in handling multiple threads concurrently.

IV. EXPERIMENTS

This section describes experiments that have been performed to validate the performance of the proposed optimization strategies for stability evaluation. Four experiments are presented that span different environmental conditions. Figure 2 shows the final assembly configuration of each example where objects are labelled with progressive integer numbers. These environments have been assembled interactively by the user. In addition to the individual optimizations two combined optimizations have also been experimented, namely $A + B$ which stands for a combined approach exploiting both prior knowledge about stable subassemblies and precedence-based optimization, and $A + B + D$ that adds clustering and parallel processing. Table I summarizes the experimental results. The computational time required to compute all the possible disassembly sequences (including those sequences that require destructive operations) is re-

Algorithm 1 Computation of disassembly subgraph of a node

```
procedure compute_disassembly_graph(X_i)
Initialize graph with node X_i
for all Nodes in the graph do
  if (!node_is_stable) then
    node_is_stable=evaluate_node_stability(current_node)
  end if
  if (node_is_stable) then
    Environment_cluster()
    if clusters_found>1 then
      for all clusters do
        compute_disassembly_graph(cluster)
      end for
      Merge clusters and update stability of nodes
    else
      Add children nodes to the graph
    end if
  else
    remove current_node from the graph
  end if
end for
```

Algorithm 2 Evaluation of node stability

```
procedure evaluate_node_stability(current_node)
node_is_stable ⇐ true
if Optimization rules (A,B) fail then
  for iteration = 0 to MAX_ITERATION do
    for all objects in current_node do
      Compute linear and angular velocity
      if (linear_velocity≥LinVel_threshold)
      or (angular_velocity≥AngVel_threshold) then
        node_is_stable ⇐ false
        return node_is_stable
      end if
    end for
    Advance step of simulation
  end for
end if
return node_is_stable
```

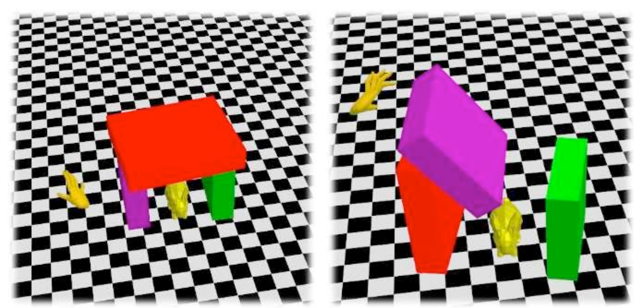

Fig. 1. Example of a stable configuration (left image) and one unstable configuration (right image).

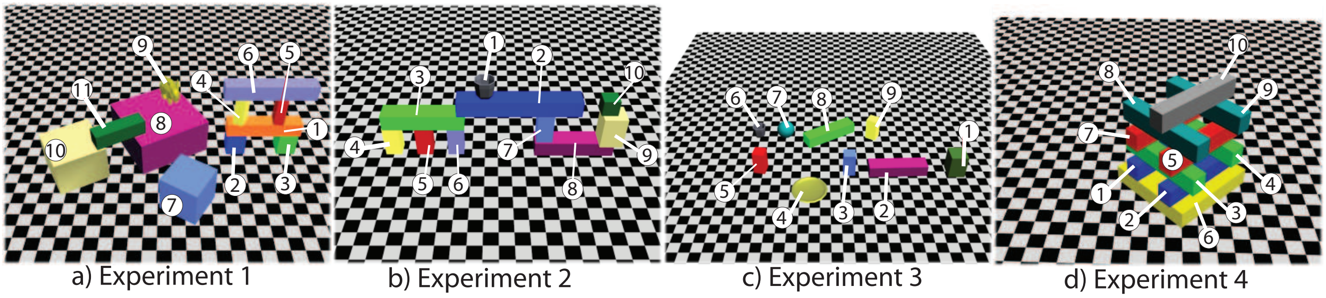

Fig. 2. User-assembled environments in 4 proposed experiments.

ported for all the optimizations. The number of physics-based disassembly attempts and the number of stable configurations that have been completely simulated is also included for each optimization. Finally, Table I reports for each experiment the speedup obtained by optimization $A + B + D$, the total number of stable configurations, the total number of disassembly sequences and the list of the provided assembly demonstrations (each demonstration is represented as a set where objects appear in the order in which they are assembled by the user).

Experiments have been run on an Intel Core 2 quad CPU (@$2.66Ghz$,$4Gb$ RAM). Experiment 1 is a complex environment comprising 11 objects. Three clusters of adjacent objects can be immediately identified at the beginning of the disassembly process ($\{1, 2, 3, 4, 5, 6\}$,{8,9,10,11},$\{7\}$). Four assembly demonstrations are provided by the user. A first observation, which holds for all the experiments, is that the computational time of each trial is essentially determined by the total number of stable configurations that have been completely simulated. Optimization B performs significantly better than optimization A. Optimization $A+B$ does not provide any significant improvement compared to B alone, thus confirming that optimization B is more general than A. Moreover, the clustering approach, in this experiment, outperforms both optimizations A and B, as only 17 configurations were required to be tested. Optimization $A + B + D$ provided the best result with a total time of $14.11s$ and only 4 fully simulated configurations. Another general observation is that parallel computation (D) provides only a limited improvement over the sequential clustering approach (C) where clusters are disassembled with a single thread. This behavior is due to the relative small size of the clusters that are identified in the proposed examples. In principle, environments with larger clusters would benefit from a parallel approach but they can not be easily tested since the total number of possible disassembly sequences would become intractable.

Experiment 2 comprises 10 objects. Four assembly demonstrations are provided by the user. The initial configuration is a single agglomerate of objects. However, if the clustering optimization is enabled after removal of objects 1 (glass) and object 2 the algorithm is able to identify two separate clusters each one made of four objects ($\{3, 4, 5, 6\}, \{7, 8, 9, 10\}$). Figure 3 shows the disassembly trees of the two clusters and the resulting merged subgraph, which is a partial subtree as well since it does not include object 1 and 2. The merged tree is automatically generated and therefore all its nodes do not have to be physically simulated. The large size of the merged tree compared to the size of its parent subgraphs highlights the high efficiency of the clustering optimization in finding stable configurations. It is also interesting to note that in this experiment optimization B by itself provides even better results than C.

The last two experiments illustrate particular environment configurations. Experiment 3 comprises 9 isolated objects lying on the ground. Four assembly demonstrations are initially provided by the user. Since all the objects are stable and they are not colliding with each other, there are 9 single-body clusters and the total number of possible disassembly sequences is given by all the possible permutations of the initial configuration ($9! = 362880$). It turns out that the clustering optimization C outperforms both A and B. Parallel processing does not improve efficiency as all the clusters contain only one object. Optimization B alone is anyhow more than three time faster than the result obtained without any optimization. It is also worthwhile noting that if a fifth demonstration is added by the user (given by the following ordered assembly sequence $\{8, 1, 4, 9, 2, 7, 6, 5, 3\}$) then the computational time of optimization B drops to $37.7s$, meaning that providing more demonstrations and hence reducing the number of precedence constraints greatly helps the optimization process. However, it must be pointed out that providing a large number of demonstrations can be demanding and time consuming for the user, whereas optimization C is performed automatically by the system without the need of user involvement.

Experiment 4 comprises 10 objects which are organized into a single cluttered cluster. The environment configuration determines a large number of physical constraints. Therefore the total number of disassembly sequences (16) as well as the disassembly time are quite low. An important observation is that the presence of a single cluster for each possible subassembly state determines that the clustering optimization is ineffective whereas optimizations A and B provide a speedup of more than two times.

A general remark that can be deduced from the previous examples is that optimizations B and C are somehow complementary since they provide positive effects in dif-

TABLE I

EXPERIMENTAL RESULTS FOR THE PROPOSED EXPERIMENTS.

Experiment	Time(seconds) for each optimization, in brackets (total disassembly attempts, fully simulated configurations)										
	None	A	B	$A+B$	C	D	$A+B+D$	speedup	stable configurations	disassembly sequences	Demonstrations (ordered assembly sequences)
1	200.53 (506,143)	152.50 (467,103)	71.84 (401,37)	70.56 (401,37)	20.28 (399,17)	17.05 (399,17)	14.11 (386,4)	14.2	143	46200	{2,3,8,9,10,1,5,11,4,6,7} {3,8,10,2,1,5,4,6,7,9,11} {8,7,2,9,10,3,11,1,5,4,6} {2,10,3,1,4,8,7,5,11,6,9}
2	109.23 (312,108)	83.31 (285,81)	34.50 (228,24)	34.31 (228,24)	39.06 (249,37)	37.62 (249,37)	20.95 (226,14)	5.2	108	8380	{5,4,6,3,8,7,2,9,1,10} {8,9,5,6,4,7,10,3,2,1} {8,6,4,5,7,3,9,2,10,1} {8,9,10,6,4,5,7,3,2,1}
3	616.16 (511,511)	565.81 (479,479)	166.61 (128,128)	157.16 (123,123)	24.12 (9,9)	23.20 (9,9)	24.09 (9,9)	25.6	511	362880	{1,2,3,4,5,6,7,8,9} {9,8,7,6,5,4,3,2,1} {1,3,5,7,9,2,4,6,8} {6,2,8,1,9,3,5,7,4}
4	14.70 (70,14)	5.78 (61,5)	5.66 (61,5)	5.5 (61,5)	14.30 (70,14)	14.42 (70,14)	5.65 (61,5)	2.6	14	16	{6,1,2,3,4,5,7,8,9,10}

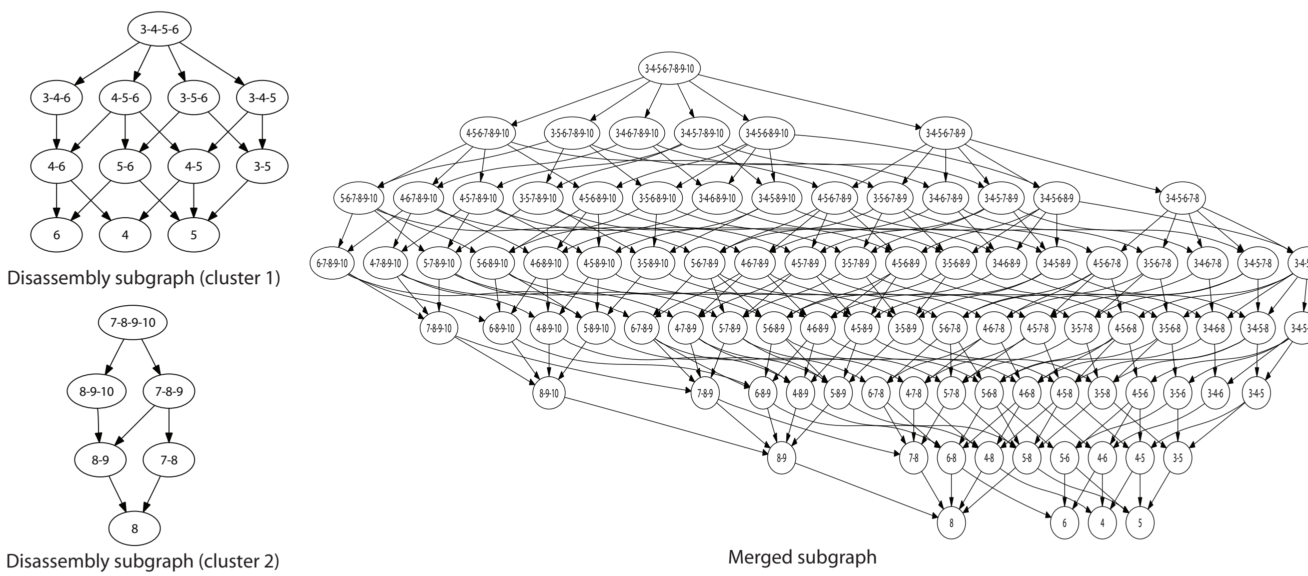

Fig. 3. Experiment 2: disassembly trees of two clusters and their resulting merged disassembly graph.

ferent environment conditions. In particular, optimization C is effective for environments where separate clusters can emerge in the disassembly phase, while optimization B is effective for constrained environments given a sufficient number of user demonstrations. The combined use of the two optimizations leads to the best performance.

V. NON-DESTRUCTIVE DISASSEMBLY PLANNING

A non-destructive disassembly planner has been developed to generate physically plausible disassembly motions. A disassembly motion of an object is non-destructive if the object can reach an infinite distance from its initial location without colliding with the remaining bodies. The input of the planner is one of the disassembly sequences identified by applying the algorithm described in section III. The planner returns the computed disassembly motions if the selected sequence of objects can be successfully disassembled in a non-destructive manner, otherwise the planner returns that a non-destructive disassembly path can not be found. The adopted motion planner is derived from the method proposed in [1]. At first, the motion planner tries to disassemble an object by applying forces along pre-computed removal directions. Removal directions are vectors sampled on the unit upward hemisphere that do not point towards bodies that are initially in contact with the object to be disassembled. Therefore, each disassembly attempt is a physics-based simulation that moves an object along a straight line. A disassembly attempt is successful if it is able to guide the object beyond a distance threshold without any collision to the other bodies in the environment. If all the straight-line disassembly attempts fail then a motion planner based on Rapidly-exploring Random Tree is invoked in order to detect feasible disassembly paths that require more complex motions. The approach is called targetless RRT (T-RRT) [1] since there is not a single goal configuration. Feasible disassembly paths are simulated in the physics-based environment by applying forces and torques to the object (kynodinamic planning approach).

An example of a non-destructive disassembly plan is presented in Figure 4. The environment consists of a caged bunny, which is the first object to be disassembled. The bunny can not be disassembled using simple removal directions, but the T-RRT is able to find a feasible path. The top left image also shows the generated random tree. The remaining objects can be disassembled using the straight line removal approach.

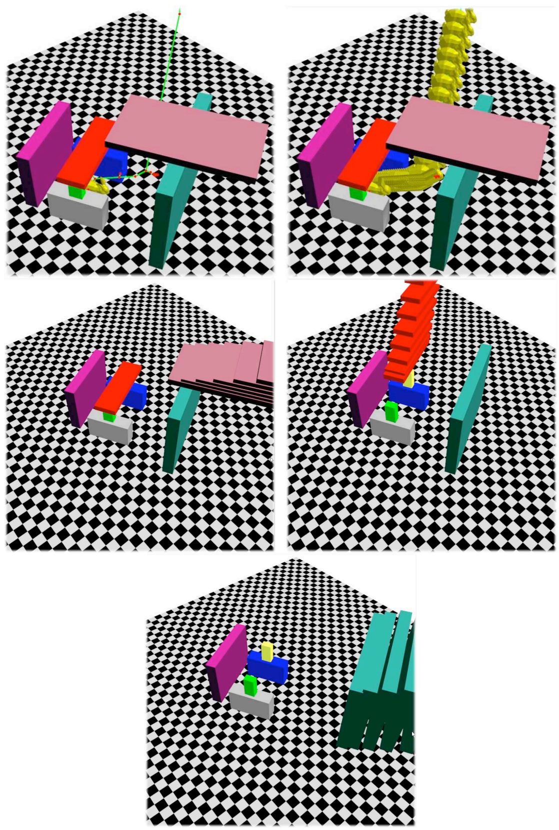

Fig. 4. A non-destructive disassembly experiment also shown in the accompanying short video. RRT-based disassembly of a caged bunny (top row) and straight-line disassembly of some of the remaining objects (second and third rows). Object trails are displayed for convenience.

VI. CONCLUSIONS

A novel approach for efficient disassembly planning of rigid bodies has been introduced. The method allows computation of all the physically stable subassembly configurations and all the possible destructive disassembly sequences of a set of objects. Optimizations based on precedence relations and geometrical clustering have been proposed. A non-destructive algorithm for computing feasible disassembly paths has also been integrated.

VII. ACKNOWLEDGMENTS

This research is partially supported by Laboratory AER-TECH of Regione Emilia-Romagna, Italy.

REFERENCES

[1] I. Aguinaga, D. Borro, and L. Matey. Parallel RRT-based path planning for selective disassembly planning. *Intl Journal of Advanced Manufacturing Technology*, 36:1221–1233, 2008.

[2] J. Aleotti and S. Caselli. Robot Grasp Synthesis from Virtual Demonstration and Topology-Preserving Environment Reconstruction. In *IEEE/RSJ Intl Conference on Intelligent Robots and Systems, (IROS)*, San Diego, USA, October 2007.

[3] K. Chen and J.M. Henrioud. Systematic Generation of Assembly Precedence Graphs. In *IEEE Intl Conference on Robotics and Automation*, pages 1476–1482, 1994.

[4] J. Cortes, L. Jaillet, and T. Simeon. Disassembly Path Planning for Complex Articulated Objects. *IEEE Transactions on Robotics*, 24(2):475–481, 2008.

[5] T. Dong, L. Zhang, R. Tong, and J. Dong. A hierarchical approach to disassembly sequence planning for mechanical product. *The International Journal of Advanced Manufacturing Technology*, 30:507–520, 2006.

[6] R. Gadh, H. Srinivasan, S. Nuggehalli, and R. Figueroa. Virtual Disassembly - A Software Tool for Developing Product Dismantling and Maintenance Systems. In *IEEE Annual RELIABILITY and MAINTAINABILITY Symposium*, 1998.

[7] D. Halperin, J.C. Latombe, and R.H. Wilson. A General Framework for Assembly Planning: The Motion Space Approach. In *ACM annual symposium on Computational geometry*, Minneapolis, USA, 1998.

[8] L.S. Homem de Mello and A.C. Sanderson. AND/OR Graph Representation of Assembly Plans. *IEEE Transactions on Robotics and Automation*, 6(2):188–199, 1990.

[9] K. Ikeuchi and T. Suehiro. Toward an assembly plan from observation, Part I: Task recognition with polyhedral objects. *IEEE Transactions on Robotics and Automation*, 10(3):368–385, 1994.

[10] S. Lee. Subassembly Identification and Evaluation for Assembly Planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3):493–503, 1994.

[11] S. Lee and F.C. Wang. Physical Reasoning of Interconnection Forces for Efficient Assembly Planning. In *IEEE Intl Conference on Robotics and Automation*, Atlanta, USA, May 1993.

[12] S. Lee and C. Yi. Subassembly Stability and Reorientation. In *IEEE Intl Conference on Robotics and Automation*, pages 521–526, 1993.

[13] A. Loomis and D. Balkcom. Computation reuse for rigid-body dynamics. In *IEEE Intl Conference on Robotics and Automation*, pages 4181–4186, 2006.

[14] R. Mattikalli, D. Baraff, and P. Khosla. Finding all Stable Orientations of Assemblies with Friction. *IEEE Transactions on Robotics and Automation*, 12(2):290–301, 1996.

[15] R.S. Mattikalli, P.K. Khosla, and Y. Xu. Subassembly Identification and Motion Generation for Assembly: a Geometric Approach. In *IEEE Intl Conference on Systems Engineering*, pages 399–403, 1990.

[16] H. Mosemann, F. Rohrdanz, and F.M. Wahl. Stability Analysis of Assemblies Considering Friction. *IEEE Transactions on Robotics and Automation*, 13(6):805–813, 1997.

[17] H. Ogata and T. Takahashi. Robotic Assembly Operation Teaching in a Virtual Environment. *IEEE Transactions on Robotics and Automation*, 10(3):391–399, jun 1994.

[18] N. S. Ong and Y. C. Wong. Automatic Subassembly Detection from a Product Model for Disassembly Sequence Generation. *The International Journal of Advanced Manufacturing Technology*, 15:425–431, 1999.

[19] A.D. Sappa and M.A. Garcia. Hierarchical Clustering of 3D Objects and its Application to Minimum Distance Computation. In *IEEE Intl Conference on Robotics and Automation*, pages 5287–5292, April 2004.

[20] S. Sundaram, I. Remmler, and N.M. Amato. Disassembly Sequencing Using a Motion Planning Approach. In *IEEE Intl Conference on Robotics and Automation*, pages 1475–1480, 2001.

[21] F. Torres, S.T. Puente, and R. Aracil. Disassembly Planning Based on Precedence Relations among Assemblies. *Intl Journal of Advanced Manufacturing Technology*, 21:317–327, 2003.

[22] J.J. Waarts, N. Boneschanscher, and W.F. Bronsvoort. A Semi-Automatic Assembly Sequence Planner. In *IEEE Intl Conference on Robotics and Automation*, pages 2431–2438, 1992.

[23] R. Zöllner, M. Pardowitz, S. Knoop, and R. Dillmann. Towards Cognitive Robots: Building Hierarchical Task Representations of Manipulations from Human Demonstration. In *IEEE Intl Conference on Robotics and Automation*, pages 1535–1540, 2005.