2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

Cut Sequencing Algorithm for Safely Disassembling Large Structures

James Akl, Sumanth Pericherla, and Berk Calli

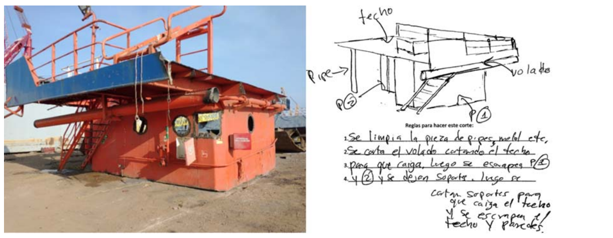
Abstract— **Disassembly and fragmentation are key operations in the dismantling and recycling of decommissioned structures such as aircraft, vessels, and buildings. Often, such operations are hazardous requiring careful planning for safe execution based on the experience and intuition of workers and forepersons. We propose and devise an algorithm for the automated sequencing of cuts to disassemble large structures. Using feedback from physics-based simulations and a mathematical model for safety, our algorithm performs sequential decision-making yielding the order of the cuts on the structure and the corresponding safe standing positions of the cutter (representing a worker or a robot). Our goal is to determine a sequence of cuts and cutter locations to maximize safety for the cutter and the environment. We establish the optimal solution via exhaustive searching, and design a greedy decision scheme to reduce the search runtime. Using our evaluations in simulation, we compare our greedy decision scheme against exhaustive searching and random searching, concluding that it satisfices the goal with high safety scores and low runtime.**

I. Introduction

In light of the Circular Economy [1] and sustainable development [2], there is a growing importance of the end-of-life handling of products and structures, such as refurbishing and material recovery. In effect, the processes of disassembly, deconstruction, dismantling, and recycling are of particular interest to decommissioned vessels [3], aircraft [4], [5], buildings [6], offshore platforms [7] or other large structures. However, the disassembly of such large structures involves difficult and hazardous operations often citing concerns over occupational safety and environmental impact as in the ship recycling industry [8], [9]. The automation of such disassembly operations can help reduce the exposure to risks and hazards by improving occupational safety and decreasing the dependence on low-cost labor for dangerous tasks. For instance, these productivity and safety benefits are explored in robot-assisted building deconstruction [10].

This paper is motivated by the need to break down large structures using gas torches into smaller units as seen in shipbreaking and metal scrapyards. In these unstructured and hazardous environments, an incorrect sequence of cuts can lead to a variety of dangers such as: fragments falling on the worker; the structure tipping over due to a shifting center of gravity; the structure collapsing on the worker, among other risks. As such, workers and forepersons agree on a cutting plan prior to conducting any cuts. These plans are primarily derived from the experience and intuition of the

The authors are with the Robotics Engineering Department, Worcester Polytechnic Institute, 27 Boynton Street, Worcester, MA 01609, USA.
E-mail: {jgakl, spericherla, bcalli}@wpi.edu
This work was supported in part by funding from European Metal Recycling Ltd. via the NSF I/UCRC ROSE-HUB under grant 1939061.



Reglas para hacer este corte:
1. Se limpia la pieza de pipes, metal etc,
2. Se corta el volado cortando el techo
3. para que caiga, luego se escrapea P①
4. y ② y se dejan soportes, luego se
cortan soportes para que caiga el techo y se escrapea el techo y paredes.

Fig. 1. Example cutting plan from a shipbreaking yard prepared by a worker and foreperson to disassemble a large steel structure.

skilled workers and forepersons as well as general safety guidelines (see Fig. 1).

Nevertheless, safety planning can be improved by reducing the dependence on subjective assessment and intuition, thereby empowering workers and safety teams with automated sequencing and computational evaluation techniques. In effect, by utilizing physics-based simulations and safety modeling, the algorithmic generation of cutting sequences would enable more concrete evaluation of cutting scenarios. Thus, cutting sequences can be generated based on physical laws and can then be refined before execution via the domain expertise of workers and forepersons. We believe that our automated decision scheme is a step in the direction of improving the safety planning of structural disassembly.

In this paper, we tackle the sequential decision problem of selecting the order of cuts on input structures of varied shapes and sizes. We assume that the cut locations are determined *a priori* and provided to our algorithm, which then sequences these cuts for a safe operation (Fig. 2). Here, safety is related to protecting the cutter (a human worker or a robot) from hazards by keeping them as far away from falling segments.

In our formulation, the structure is modeled as a linkage

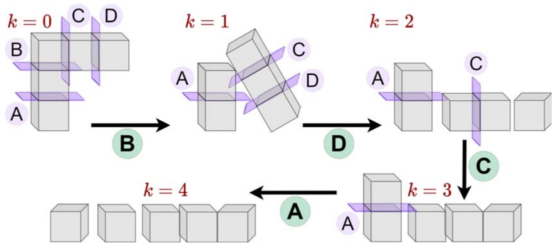


Fig. 2. Conceptual diagram illustrating the sequential decision problem of structural disassembly. The input is a partitioned object with segments and cutting locations. At each step k, the decision agent selects a cut to execute until the structure is fully-disassembled. Each choice impacts the safety of the environment whereby the resulting sequence $(\mathsf{B}, \mathsf{D}, \mathsf{C}, \mathsf{A})$ can be assessed. The goal is to find a sequence that maximizes overall safety in the environment, *i.e.*, the cutter and their surroundings.

where its segments are considered links connected to each other via (rigid) joints. Here, the action of cutting is to disconnect two links at a particular joint. Our algorithm gives a series of decisions each of which is selecting one of these joints and disconnecting it. The cuts are performed in physics-based simulations wherein the safeties of the decision outcomes are computed to provide feedback to the decision agent. The number of decisions (chosen cuts) is thus equal to the number of joints n, resulting in n steps to completely fragment the object into its constituent segments. The goal is then to choose a sequence of cuts that fully disassembles the structure while maximizing safety. Here, safety considers dynamic factors which concern the structure's physics as it falls in the environment, as well as geometric factors which concern the proximity of moving segments (*i.e.*, hazards) to the cutter's location. These constructs are precisely defined in the mathematical model developed in Section IV.

This decision problem can be stated as a combinatorial optimization, where the objective is to find the sequence maximizing safety from a finite and discrete set of cutting sequences. Here, exhaustive search scales factorially with the input size n (the number of choices, *i.e.*, joints) becoming intractable for larger inputs. As such, we develop a greedy decision search to prune large parts of the search space and satisfice the problem's goal.

The core novel contributions of this work are:

- Devising an algorithm for the sequencing of cuts on partitioned structures for safely disassembling them.
- Formulating a mathematical model for safety that incorporates both dynamic and geometric factors.
- Evaluating the decision agent's performance and runtime in simulation against diverse test objects.

We emphasize our assumption that the input structure is partitioned *a priori* into constituent segments. This can be accomplished by either manually partitioning the object (in practice, based on the experience of workers and safety forepersons as in Fig. 1), or by utilizing a volumetric partitioning algorithm such as [11] which is skeleton-based, [12] which is search-based, or [13] which is convexity-based. We instead focus on developing a decision scheme for algorithmically disassembling such partitioned structures while maximizing safety for the environment and the cutter. In turn, the cutter's ideal location is computationally prescribed at each step depending on the state and on the cutting tool's reach. We elaborate the specifics of our modeling and procedures in Sections III–IV.

II. Related Work

In this section, we review existing work in sequencing algorithms and approaches for product disassembly as well as planning methods for building deconstruction.

A. Disassembly Sequence Planning

There is extensive work in disassembly sequence planning for the end-of-life handling of industrial products. [14] reviews the recent developments of robotic applications in product disassembly, distinguishing between predefined disassembly processes and more adaptable and flexible disassembly schemes. Many optimization-based approaches are applied for the disassembly process on general product structures. For instance, [15] uses multi-objective optimization to maximize parallelism, ergonomics, workload balancing, while minimizing disassembly time and product rotation count. [16] formulates the problem as an extended AND-OR graph while considering practical constraints such as reuse probability and environmental impacts. [17] represents the problem as a precedence graph and minimizes the total disassembly cost via integer programming. Other optimization schemes incorporate human-robot collaboration [18], [19]. Elsewhere, evolutionary [20] and genetic algorithms [21], [22] are applied extensively for multi-objective optimization under constraints. Hybrid approaches are also observed such as using genetic algorithms and AND-OR graphs [23], using genetic algorithms and fuzzy logic [24], or integrating several cognitive functions with a knowledge base [25]. In addition a wide variety of Petri net representations are used to model the process and resolve it using optimization-based approaches [26], [27] or fuzzy inference [28]. In addition, more specialized methods are tailored to a particular range of products such as electronics [29]–[31].

While these approaches address sequence generation for disassembling a variety of product structures, they do not consider the transient effects of the structure's disassembly and its impacts on the safety of its surroundings. In part, this is due to the structured setting of the product's disassembly and the comparatively smaller scale of the objects (*e.g.*, consumer electronics). Instead, our algorithm targets the breakdown of large structures, which expose the cutter and surrounding environment to hazards. To the best of our knowledge, we present the first work that targets safety maximization during disassembly by considering the dynamic and geometric factors of the structure's breakdown.

B. Building Deconstruction Planning

A related area of research is the deconstruction of buildings and its planning. [32] surveys many aspects related to the life-cycle of buildings and mentions uses and examples of automated planning for deconstruction. In addition, [33] reviews key problems associated with building demolition and the opportunities for automation to mitigate their effects. In the case of planning, multi-objective optimization schemes are applied with great variety for selective disassembly planning [34] and deconstruction strategy planning [35]. For automated prefabrication, [36] presents a method comparing source and target structure configurations with known parts to sequence their disassembly and reassembly. Such deconstruction plans contrast with more lower-level task planning as in robot-assisted deconstruction [10], [37] and refurbishing [38].

In recent developments, Building Information Modeling (BIM) systems are exploited to obtain structured representations of building parameters for deconstruction planning [39]–[43] and for deconstruction waste manage-

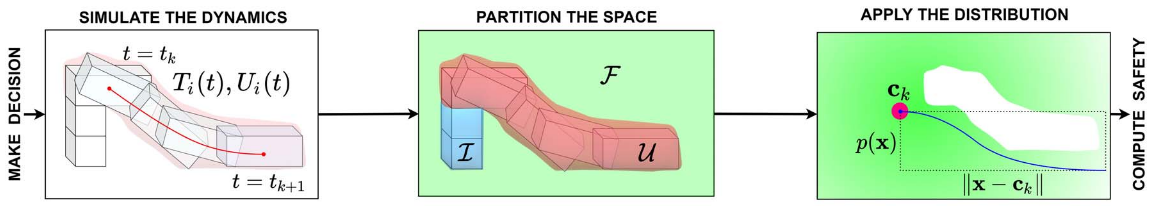


Fig. 3. Illustrating the computational elements of the safety model. A decision a_k is simulated wherein motions begin at t_k and end at t_{k+1}. The environment space $\mathbb{R}^3$ is partitioned into unsafe ($\mathcal{U}$), immobile ($\mathcal{I}$), and free ($\mathcal{F}$). The safety model considers both dynamic effects (aggregate motion of the structure) and geometric effects (proximity of moving segments to the cutter position $\mathbf{c}_k$). The cutter's ideal position $\mathbf{c}_k$ is computed algorithmically.

ment [44]. However, BIM representations require considerable explicit modeling and are available mostly for newer and highly-organized construction projects. In contrast, many deconstruction projects involve aging structures which lack a BIM representation. Moreover, many large structures are not buildings and are incompatible with the semantics of BIM systems. In effect, 3D imaging [45] is a flexible alternative to obtaining structure representations.

In many of these planning techniques, building deconstruction is tackled statically without considering the kinetics of falling fragments. This is due to the highly-structured and highly-regulated nature of the construction industry where it can be assumed that mechanized operations can safely execute the disassembly plans via established procedures. Additionally, in the case of BIM, disassembly plans can be generated with great precision since much of the building components and parameters would be explicitly known. In contrast, scrapyard environments are highly unstructured and hazardous requiring workers to thermally cut structures using gas torches. Here, the kinetics of structural disassembly are crucial for occupational safety. To the best of our knowledge, we present the first such algorithm to maximize safety during structural disassembly.

III. Problem Formulation

Our structural disassembly problem involves the following elements: (1) the environment (3D space) containing the partitioned input structure, the cutter's position, and the available decisions, *i.e.*, the remaining cutting locations on the structure; (2) the dynamics of the environment given the chosen decisions; and (3) the safety function to score decisions. We recall that the structure is partitioned into n_{links} segments. Its topology can be represented as a kinematic chain with n_{links} links (the segments) and n joints (the cutting locations). In addition, the physical properties of each link are known (and configured in simulation).

Formally, the environment state at decision step k is s_k which describes the object's state and the cutter's position $\mathbf{c}_k \in \mathbb{R}^3$. Here, the object's state encodes the geometric and dynamic information of each structure segment at step k. This would include, for instance, each segment's shape, size, pose, mass, and so on. The state s_k thus describes at step k the properties of each segment and the cutter's location $\mathbf{c}_k$. A decision is to select and disconnect one of the structure's joints. We denote the decision at step k as $a_k \in \mathcal{A}_k$. Here, $\mathcal{A}_k$ is the set of available decisions, *i.e.*, the joints that have yet to be disconnected. Accordingly, a decision sequence can be expressed as $\mathbf{a} = (a_k)_{k=0}^{n-1} = (a_0, \ldots, a_{n-1})$. Note that the decision sequence bijectively maps the decision steps k to the joint indices j, that is, $\mathbf{a} : \{k\}_{k=0}^{n-1} \leftrightarrow \{j\}_{j=0}^{n-1}$.

The dynamics of the environment can thus be expressed as $s_{k+1} = f(s_k, a_k)$ where f maps to the next state s_{k+1} given the decision a_k taken at the current state s_k. In our case, f is a physics-based simulation where selecting a cut (decision a_k) at a particular scene (current state s_k) leads to motions in the scene. Once all entities in the simulation cease to move, the next state s_{k+1} (*i.e.*, the resultant scene) is realized. This is in agreement with safety practices in the scrapyard, where cuts are performed on the structure after it stabilizes and its pieces cease to move. Now, the safety of a decision can be measured using a safety function $S : \mathcal{A}_k \to [0, 1]$ that maps from decisions to the unit interval. Specifically, the safety function scores the decision a_k based on its outcome and transition to s_{k+1}, *i.e.*, $S(a_k)$ is a function of s_{k+1} and the transition from s_k to s_{k+1}. We now define the safety of a decision sequence $\mathbf{a}$ as the weighted geometric mean of its individual decision safeties $S(a_k)$, as in: $S(\mathbf{a}) = \prod_{k=0}^{n-1} S(a_k)^{\lambda_k} = S(a_0)^{\lambda_0} \cdots S(a_{n-1})^{\lambda_{n-1}}$.

The weights λ_k are expressed as a proportion, *i.e.*, $\lambda_k \in [0, 1]$ and $\sum_{k=0}^{n-1} \lambda_k = 1$. These weights λ_k are assigned such that the worst-case decision $a_{\text{wc}} = \operatorname{argmin}_{a_k \in \mathbf{a}} S(a_k)$ has a corresponding weight of λ_{wc} whereas all remaining weights $\lambda_{k \neq \text{wc}} = \frac{1-\lambda_{\text{wc}}}{n-1}$ are uniform. By setting $\lambda_{\text{wc}} = \frac{1}{2}$, the sequence safety is penalized by the worst-case decision irrespective of the sequence length. In doing so, the sequence safety can assess and compare sequences of any length yet remains skewed by its most dangerous decision. By design, this multiplicative formulation heavily penalizes the sequence safety for unsafe decisions. For instance, one strictly unsafe decision with $S(a_k) = 0$ would result in a strictly unsafe sequence $S(\mathbf{a}) = 0$. This is a desirable modeling choice due to the sequential nature of the problem (non-episodicness) given the permanent harm inflicted by potential hazards. Note that λ_{wc} quantifies the weight of the worst-case decision, meaning that a higher λ_{wc} leads to a more conservative measure of sequence safety.

Finally, the goal is to maximize the sequence's safety:

$$\mathbf{a}^* = \operatorname{argmax}_{\mathbf{a}} S(\mathbf{a}) = \underset{a_0,\ldots,a_{n-1}}{\operatorname{argmax}} \prod_{k=0}^{n-1} S(a_k)^{\lambda_k} \qquad (1)$$

In the next sections, we define our safety model S as well as our decision algorithm to determine the cutter's position $\mathbf{c}_k$ and the decisions a_k.

IV. Safety Model

The safety $S(a_k)$ of a decision a_k is modeled in such a way to capture the dynamic and geometric outcomes of the transition from s_k to s_{k+1}, This starts when the cut is made and stops when all segments in the scene stop moving.

We define the safety $S(a_k)$ as:

$$S(a_k) = S_d(a_k)\, S_g(a_k) \tag{2}$$

where $S_d(a_k)$ is the dynamic safety and $S_g(a_k)$ is the geometric safety. We note that each of S, S_d, and S_g map decisions to the unit interval, *i.e.*, $\mathcal{A}_k \to [0, 1]$.

By design, S_d captures the kinetic outcomes of cutting the structure, *i.e.*, the 'intensity' of the segments' motion. In contrast, S_g captures the kinematic outcomes of cutting the structure, *i.e.*, the 'closeness' of the segments' paths as they fall relative to the cutter's position. These modeling choices are based on the following assertions for decision safety:

- The decision is safer when the aggregate motion of the segments in the scene has a lower magnitude.
- The decision is safer when the aggregate traversal of the segments in the scene is further from the cutter's position.

Both S_d and S_g are formally defined below accordingly.

A. Dynamic Safety

While there are many ways to express the magnitude of the segments' aggregate motion, we wish to concisely capture inertial effects during motion. As such, a straightforward choice is to use the kinetic energies of each segment.

We first define the time interval $t \in [t_k, t_{k+1}]$ wherein the transition from s_k to s_{k+1} takes places and the segments start ($t = t_k$) and stop moving ($t = t_{k+1}$). Now, we define $T_i(t)$ and $U_i(t)$ to be the kinetic and gravitational potential energies of each segment $i \in \{1, n_{\textsf{links}}\}$ as well as their totals $T(t) = \sum_i T_i(t)$ and $U(t) = \sum_i U_i(t)$. We note that the total mechanical energy $T(t) + U(t)$ in the scene is conserved for $t \in [t_k, t_{k+1}]$ since dissipative forces (*e.g.*, friction, air resistance) are negligible. As such, we can define the normalized energies $\tilde{T}(t) = \frac{T(t)}{T(t)+U(t)}$ and $\tilde{U}(t) = \frac{U(t)}{T(t)+U(t)}$. These capture the ratios of kinetic and potential energies respectively. Stated differently, they capture the instantaneous ratio of mechanical energy due to motion and to non-motion.

We would like S_d to increase with a larger ratio due to non-motion, *i.e.*, when the structure is disassembled more gently. We thus define dynamic safety as follows:

$$S_d(a_k) = \min_{t\in[t_k, t_{k+1}]} \tilde{U}(t) \tag{3}$$

As S_d is the minimum of $\tilde{U}(t)$ across the transition from s_k to s_{k+1}, Eq. (3) yields the desired properties of mapping to $[0, 1]$ and of decreasing with higher aggregate motion. The relationship between the energies ($\tilde{U}$, $\tilde{T}$) and the decision a_k is implicit whereby the decision's outcome determines the energy values through the aforementioned manner.

B. Geometric Safety

In addition to dynamic safety, we wish to consider the proximity of the segments' traversal relative to the cutter's location during the transition interval $t \in [t_k, t_{k+1}]$.

For this, we partition the environment space $\mathbb{R}^3$ into three sets (Fig. 3): the unsafe space $\mathcal{U}$, the immobile space $\mathcal{I}$, and the free space $\mathcal{F}$. These three spaces are defined as follows.

The unsafe space $\mathcal{U}$ contains all points visited by moving object segments during the transition interval $t \in [t_k, t_{k+1}]$ after a decision a_k. This can be expressed as $\mathcal{U} = \bigcup_{t\in[t_k,t_{k-1}]} V(t)$ where $V(t) = \bigcup_{i=1} V_i(t)$ is the set of points contained within all moving segments at time t and $V_i(t)$ is the set of points contained within the moving segment i at time t. Thus, $\mathcal{U}$ represents the collision set in the environment containing all points visited by moving segments throughout their motions.

The immobile space $\mathcal{I}$ contains the points of all segments that remain immobile throughout the transition interval $t \in [t_k, t_{k+1}]$ after a decision a_k. These are the stationary segments, which are not hazards. The free space $\mathcal{F}$ contains the points that remain unoccupied throughout the transition interval $t \in [t_k, t_{k+1}]$ after a decision a_k. These are safe and empty locations that have not experienced collisions.

Using these definitions, we distinguish the unsafe space $\mathcal{U}$ from the safe space $\mathcal{S} = \mathcal{U}^{\complement} = \mathcal{F} \cup \mathcal{I}$. As such, the indicator function $\mathbf{1}_{\mathcal{S}} : \mathbb{R}^3 \to \{0, 1\}$ partitions the environment into unsafe positions where $\mathbf{1}_{\mathcal{S}}(\mathbf{x}) = 0$ and safe positions where $\mathbf{1}_{\mathcal{S}}(\mathbf{x}) = 1$. This safety indicator function expresses the safety of a position in the environment.

We algorithmically determine an ideal position $\mathbf{c}_k$ for the cutter as described in Section V. This $\mathbf{c}_k$ represents the prescribed location that the cutter should cut from. In actuality, the cutter may stray from their ideal position and we thus represent their position as a random variable $\mathbf{x} \in \mathbb{R}^3$. We model $\mathbf{x}$ to follow a unimodal and symmetric distribution centered at its mode (the cutter's ideal position $\mathbf{c}_k$) and with radial decay. The intent is to decrease the probability of the cutter's presence as $\|\mathbf{x} - \mathbf{c}_k\|_2$ grows. For this, a trivariate Gaussian distribution with mean $\mathbf{c}_k$ and covariance matrix $r^3\mathrm{I}_3$ is appropriate. With $\mathbf{x} \sim \mathcal{N}(\mathbf{c}_k, r^3\mathrm{I}_3)$, we obtain its probability density function $p(\mathbf{x})$ by evaluating the multinormal PDF with mean $\mathbf{c}_k$ and covariance $r^3\mathrm{I}_3$.

$$p(\mathbf{x}) = \frac{1}{r^3\sqrt{(2\pi)^3}} \exp\left[-\frac{1}{2r^2}\|\mathbf{x} - \mathbf{c}_k\|_2^2\right] \tag{4}$$

Finally, define the geometric safety $S_g(a_k)$ as:

$$S_g(a_k) = \mathbb{E}[\mathbf{1}_{\mathcal{S}}(\mathbf{x})] = \int_{\mathbf{x}\in\mathbb{R}^3} \mathbf{1}_{\mathcal{S}}(\mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{5}$$

In this manner, $S_g(a_k)$ represents the cutter's expected safety in the environment after decision a_k given their ideal position $\mathbf{c}_k$. Also since $\mathbf{1}_{\mathcal{S}}(\mathbf{x}) \in \{0, 1\}$, S_g maps to $[0, 1]$ as desired. Moreover, the parameter r can be interpreted as a 'radius' wherein the cutter's safety is most crucial. A smaller r decays the safety scores rapidly prioritizing the safety nearest to the prescribed position. Conversely, a larger r decays the safety

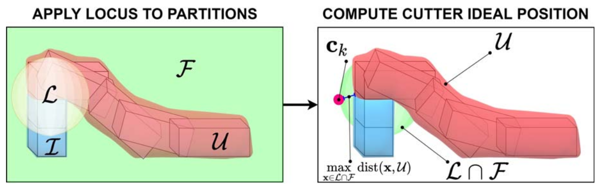

APPLY LOCUS TO PARTITIONS
COMPUTE CUTTER IDEAL POSITION
$\mathcal{F}$
$\mathcal{L}$
$\mathcal{I}$
$\mathcal{U}$
$\mathbf{c}_k$
$\mathcal{U}$
$\max_{\mathbf{x}\in\mathcal{L}\cap\mathcal{F}} \operatorname{dist}(\mathbf{x},\mathcal{U})$
$\mathcal{L}\cap\mathcal{F}$

Fig. 4. The cutter's ideal position is computed as the locus point in the free space ($\mathbf{x} \in \mathcal{L} \cap \mathcal{F}$) that is maximally-distant from the unsafe space $\mathcal{U}$.

scores more slowly giving added consideration to regions further away from the prescribed position.

In implementation, the formulation is discretized as follows: the segments' kinematic and kinetic information are sampled from the simulation as time-series data; the set $\mathcal{S}$ and its indicator function $\mathbf{1}_{\mathcal{S}}$ are implemented as a binary voxel grid; and the Eqs. (3) and (5) are approximated using discrete sums.

V. Decision Algorithm

With the safety function S defined and the decision goal $(\mathbf{a}^*, S^*)$ established in Eq. (1), we now describe the procedures and schemes used to solve the decision problem.

A. *Cutter's Ideal Position*

The decision agent not only generates a disassembly sequence, but also specifies the cutter's prescribed position for each cut (Fig. 4). In effect, the safety model requires this position $\mathbf{c}_k$ as seen in Eq. (4). We emphasize the advantage of simulating the environment dynamics: the ability to observe the cut's outcome first and then decide where the best position to cut from should have been. In this way, the cutter's ideal position is determined *a posteriori*. Specifically, the decision agent attempts and simulates a cut a_k, measures its outcome s_{k+1}, and then determines $\mathbf{c}_k$, *i.e.*, the safest position to cut from. Here, we specify a procedure to compute $\mathbf{c}_k$ given the decision a_k and its outcome s_{k+1}.

```
define ComputeIdealPosition($a_k, s_{k+1}, m$):
    $(\mathcal{F},\mathcal{I},\mathcal{U}) \leftarrow$ GetPartitions($s_{k+1}$)
    $\mathcal{L} \leftarrow$ GenerateLocus($a_k, m$)
    $\mathbf{c}_k \leftarrow \operatorname*{argmax}_{\mathbf{x}\in\mathcal{L}\cap\mathcal{F}} \operatorname{dist}(\mathbf{x},\mathcal{U})$
    return $\mathbf{c}_k$
```

After the cut, we retrieve from s_{k+1} the space's partitioning into unsafe ($\mathcal{U}$), immobile ($\mathcal{I}$), and free ($\mathcal{F}$) as defined in Section IV-B. The cutter's position locus $\mathcal{L}$ contains all candidate ideal positions and depends on the cutting margin $m > 0$ defined by the cutting tool's length. For instance, the cutting torch used in shipbreaking is around $2\,\mathrm{m}$ long, giving the cutter a margin of $m = 2\,\mathrm{m}$. We model the cutter's locus as a sphere centered at the joint location of a_k with radius m. We wish to position the cutter in the free space furthest from the unsafe space. This is expressed as the point in $\mathcal{L}\cap\mathcal{F}$ with maximum distance from $\mathcal{U}$, or $\operatorname{argmax}_{\mathbf{x}\in\mathcal{L}\cap\mathcal{F}} \operatorname{dist}(\mathbf{x},\mathcal{U})$. We do not consider the case of segments so large that $\mathcal{L}\cap\mathcal{F} = \varnothing$ wherein the cutter cannot safely reach the joint within a radius m. In such cases, the object must be re-partitioned in a reasonable manner to enable safer cutting. In a discretized implementation, one can rank the candidates by distance and select the furthest that is feasible in terms of cutting ergonomics. Now, a decision a_k can be evaluated as follows.

```
define EvaluateDecision($a_k$):
    $s_{k+1} \leftarrow$ SimulateDecision($a_k$)
    $\mathbf{c}_k \leftarrow$ ComputeIdealPosition($a_k, s_{k+1}, m$)
    $S_k \leftarrow$ ComputeSafety($a_k, \mathbf{c}_k, r$)
    return $(S_k, \mathbf{c}_k)$
```

The procedure SimulateDecision represents the physics simulator which loads the environment containing the input paritioned model and computes the outcomes of the cut a_k. The procedure ComputeSafety implements Eqs. (2)–(5).

B. *Decision Search Schemes*

We develop three alternative searches: exhaustive (ES), greedy (GS), and random (RS). For each search, the input is the partitioned object with n joints and the output is the solution $(\mathbf{a}, S, \mathbf{C})$ containing the decision sequence $\mathbf{a}$, its safety S, and the sequence $\mathbf{C}$ of ideal positions. In addition the parameters r (see Section IV-B) and m (see Section V-A) are specified based on application requirements.

ES finds the optimal solution $\mathbf{a}^*$ of Eq. (1). Both ES and RS provide a basis of comparison for computational complexity and safety performance. GS approximates $\mathbf{a}^*$ by optimizing locally $a_k = \operatorname{argmax}_{a\in\mathcal{A}_k} S(a_k)$. The schemes are specified in Algorithms 1–3. We note that $\mathcal{A}$ is the set of all decision sequences. These are permutations (without replacement) of the structure's n joints yielding a cardinality of $n!$ sequences. We recall that $\mathcal{A}_k$ is the set of remaining decisions available at step k. The cardinality of $\mathcal{A}_k$ is $n - k$ and its contents depend on $\mathcal{A}_{k-1}$ and a_{k-1} (for $k > 0$).

Algorithm 1: Exhaustive Decision Search

```
*Initialize safety for the solution sequence:* $S \leftarrow 0$
*Iterate over all possible decision sequences.*
for all $\mathbf{a}_{\mathsf{cand}} \in \mathcal{A}$ do
    *Iterate over each decision per sequence.*
    for each $a_k \in \mathbf{a}_{\mathsf{cand}}$ do
        $(S_k, \mathbf{c}_k) \leftarrow$ EvaluateDecision($a_k$)
    *Compute the candidate solution components.*
    $S_{\mathsf{cand}} \leftarrow \prod_{k=0}^{n-1} S_k^{\lambda_k}$,    $\mathbf{C}_{\mathsf{cand}} \leftarrow (\mathbf{c}_0, \ldots, \mathbf{c}_{n-1})$
    *Accept candidate solution upon improvement.*
    if $S_{\mathsf{cand}} > S$ then
        $(\mathbf{a}, S, \mathbf{C}) \leftarrow (\mathbf{a}_{\mathsf{cand}}, S_{\mathsf{cand}}, \mathbf{C}_{\mathsf{cand}})$
return $(\mathbf{a}, S, \mathbf{C})$
```

C. *Asymptotic Performance*

We express the computational complexity as the total decision evaluations $\mathsf{C}(n)$ for an input size n (number of decisions). A decision evaluation refers to a call to the EvaluateDecision routine which involves simulating the decision, computing its cutter's ideal position, and computing its safety. ES as indicated in Algorithm 1 performs

Algorithm 2: Greedy Decision Search

```
*Iterate over each decision step of the solution.*
for $k \leftarrow 0, \ldots, n-1$ do
    *Initialize safety for the current decision:* $S_k \leftarrow 0$
    *Iterate over all currently available decisions.*
    for all $a_{\mathsf{cand}} \in \mathcal{A}_k$ do
        $(S_{\mathsf{cand}},\ \mathbf{c}_{\mathsf{cand}}) \leftarrow$ EvaluateDecision$(a_{\mathsf{cand}})$
        *Accept candidate decision upon improvement.*
        if $S_{\mathsf{cand}} > S_k$ then
            $(a_k, S_k, \mathbf{c}_k) \leftarrow (a_{\mathsf{cand}}, S_{\mathsf{cand}}, \mathbf{c}_{\mathsf{cand}})$
*Compute the solution's components.*
  $\mathbf{a} \leftarrow (a_0, \ldots, a_{n-1}),\quad S \leftarrow \prod_{k=0}^{n-1} S_k^{\lambda_k}$
  $\mathbf{C} \leftarrow (\mathbf{c}_0, \ldots, \mathbf{c}_{n-1})$
return $(\mathbf{a}, S, \mathbf{C})$
```

Algorithm 3: Random Decision Search

```
*Iterate over each decision step of the solution.*
for $k \leftarrow 0, \ldots, n-1$ do
    *Randomly select a currently available decision.*
      $a_k \leftarrow \mathrm{rand}\,\mathcal{A}_k$
      $(S_k,\ \mathbf{c}_k) \leftarrow$ EvaluateDecision$(a_k)$
*Compute the solution's components.*
  $\mathbf{a} \leftarrow (a_0, \ldots, a_{n-1}),\quad S \leftarrow \prod_{k=0}^{n-1} S_k^{\lambda_k}$
  $\mathbf{C} \leftarrow (\mathbf{c}_0, \ldots, \mathbf{c}_{n-1})$
return $(\mathbf{a}, S, \mathbf{C})$
```

$n \times n!$ evaluations. Even with efficient implementation, *e.g.*, memoization (storing reusable states), these reduce to $\mathsf{C}(n) = \sum_{k=0}^{n-1} \frac{n!}{(n-k-1)!} \in \mathcal{O}(n!)$ evaluations. GS runs $\mathsf{C}(n) = \sum_{k=0}^{n-1}(n-k) = \frac{n(n+1)}{2} \in \mathcal{O}(n^2)$ evaluations and RS runs $\mathsf{C}(n) = n \in \mathcal{O}(n)$ evaluations.

VI. Evaluation in Simulation

We implement our decision environment using the simulator Gazebo [46] to evaluate our decision agent's performance using each of the search schemes against different inputs.

A. Simulation Environment

In our simulations, the object segments are considered rigid bodies connected as a kinematic chain. A cut is implemented as an instantaneous disconnection between two segments. Our simulations model rigid-body dynamics, gravitational and frictional forces, and collisions. A simulated experiment consists of loading the input object, executing cuts per the decision agent, and updating the environment's state until a solution for the object's full disassembly is chosen. We test our implementation on the 12 partitioned objects (Fig. 5). We illustrate the decision outputs of each decision scheme against the object 5B in Table I and in Fig. 6.

B. Simulation Results

As can be seen in Fig. 6, ES and GS yield similar decisions for object 5B differing only in their second and third decisions which are permuted. In contrast, RS can

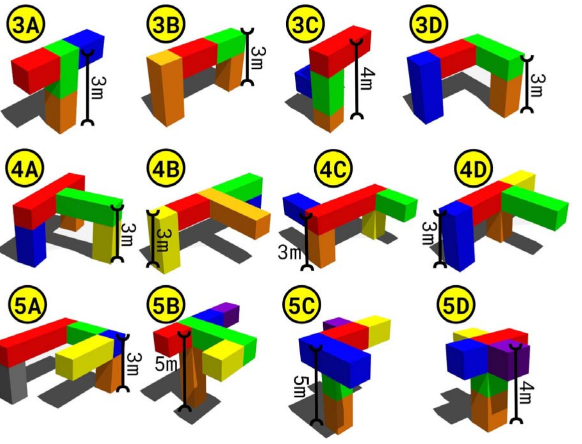


Fig. 5. The 12 partitioned test objects are shown with their label indicating the number of cuts (*e.g.*, 3A has $n = 3$) and their structural height.

TABLE I
Outputs for object 5B showing the sequence and its safeties

Scheme	Sequence $\mathbf{a}$	Decision Safeties S_k
Exhaustive	(4, 2, 3, 5, 1)	(0.79, 0.76, 0.78, 0.63, 0.65)
Greedy	(4, 3, 2, 5, 1)	(0.79, 0.76, 0.71, 0.62, 0.64)
Random	(2, 1, 5, 3, 4)	(0.79, 0.28, 0.74, 0.48, 0.99)

yield highly-dangerous decisions where large chunks of the structure tumble dynamically. In effect, the safeties of its worst decisions are $S_1 = 0.28$ and $S_3 = 0.48$.

More generally, we summarize in Table II the decision agent's performance for each search scheme against each test structure. We note the safety of each scheme's solution $S(\mathbf{a})$ and its search cost $\mathsf{C}(n)$ measured as the number of decision evaluations (see Section V-C). For example, ES computes for object 3A the optimal sequence $\mathbf{a}^*$ with safety $S(\mathbf{a}^*) = 0.7116$ using $\mathsf{C}(3) = 18$ decision evaluations. Similarly, GS computes the greedy solution $\mathbf{a}^\dagger$ for object 3A and its safety $S(\mathbf{a}^\dagger) = 0.6989$. The random solution is uniformly-sampled, and therefore we compute its expected safety and worst-case safety, which for object 3A are $\mathbb{E}[S(\mathbf{a})] = 0.6776$ and $S(\mathbf{a}_{\mathsf{wc}}) = 0.6495$ where $\mathbf{a}_{\mathsf{wc}} = \mathrm{argmin}_{\mathbf{a}\in\mathcal{A}} S(\mathbf{a})$.

We simulate all $n!$ sequences for each n-segment structure in order to compute the safety $S(\mathbf{a}^*)$ of the optimal sequence, the expected safety $\mathbb{E}[S(\mathbf{a})]$ of the uniformly sampled sequence, and the safety $S(\mathbf{a}_{\mathsf{wc}})$ of the worst-case sequence.

C. Discussion of the Results

We directly compare in Table III the performance of GS against that of ES and RS. We compare the greedy solution against the optimal solution via the ratio of their safeties $\frac{S(\mathbf{a}^\dagger)}{S(\mathbf{a}^*)}$. Furthermore, we compute the relative improvement of the greedy solution's safety against that of the random solution's expected and worst-case safeties. For our 12 test objects, the greedy solution $\mathbf{a}^\dagger$ yields a safety that is on average 96.72% of the optimal sequence's safety $S(\mathbf{a}^*)$. In some cases, the greedy solution is near optimal where $\frac{S(\mathbf{a}^\dagger)}{S(\mathbf{a}^*)} > 99\%$ such as with objects $\{$3B, 3C, 4B, 4C, 5B$\}$.

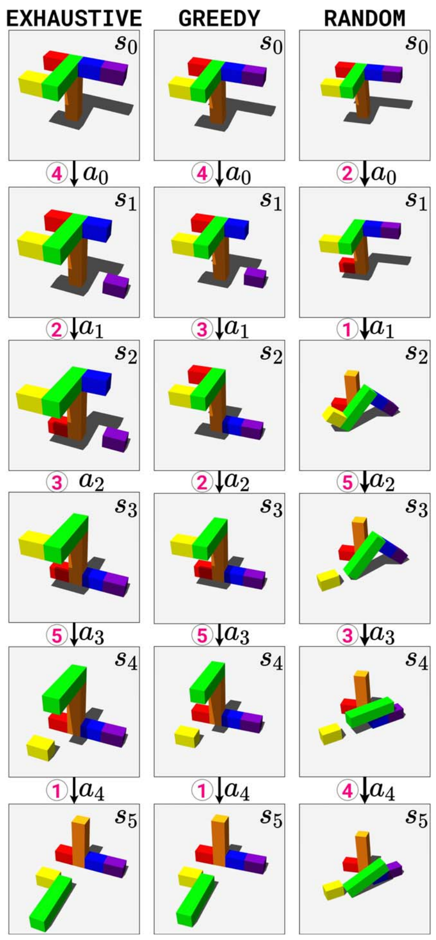


Fig. 6. The decision outcomes of each search scheme is demonstrated on object 5B. We note the difference in safety outcomes between the random scheme and the other schemes. The safety scores can be found in Table I.

Moreover, the greedy solution $\mathbf{a}^\dagger$ performs on average 6.10% better than uniformly-sampled random searching and 17.83% better than the worst-case sequence, the latter of which would yield potential risks and hazards. In addition, our results reveal that some objects are inherently safer to cut where decisions weakly impact the safety outcomes, *e.g.*, for object 4A the solution $\mathbf{a}^\dagger$ is only 0.05% safer than random and 1.10% safer than the worst-case sequence. Conversely, some objects are inherently much more dangerous to cut (4C, 5A, 5B, 5C) where decisions greatly impact the safety outcomes, *e.g.*, for object 5B the solution $\mathbf{a}^\dagger$ is 18.12% safer than random and 33.64% safer than the worst-case sequence. Our results suggest that GS yields significantly safer decisions than RS, sometimes achieving near-optimal safety using only $\mathcal{O}(n^2)$ decision evaluations. For illustration with $n = 5$ as in object 5B (Fig. 6), the GS solution is found in 15 evaluations, instead of 600 as with the ES solution.

VII. Conclusion

We formalize the safe disassembly of large structures as a sequential decision problem for which we develop an algorithm to maximize the safety for the cutter and the surrounding environment. By simulating a decision's outcomes, the agent partitions the environment into safe

TABLE II

Simulation results of each scheme's cost and performance

Input	**Exhaustive**		**Greedy**		**Random**		
(Obj, n)	C(n)	$S(\mathbf{a}^*)$	C(n)	$S(\mathbf{a}^\dagger)$	C(n)	$\mathbb{E}[S(\mathbf{a})]$	$S(\mathbf{a}_{\text{wc}})$
(3A, 3)	18	0.7116	6	0.6989	3	0.6776	0.6495
(3B, 3)	18	0.8402	6	0.8378	3	0.8150	0.7505
(3C, 3)	18	0.8104	6	0.8097	3	0.7561	0.7149
(3D, 3)	18	0.7628	6	0.7270	3	0.7180	0.6470
(4A, 4)	96	0.9226	10	0.9050	4	0.9045	0.8950
(4B, 4)	96	0.8053	10	0.7980	4	0.7268	0.6480
(4C, 4)	96	0.7934	10	0.7858	4	0.7402	0.5661
(4D, 4)	96	0.8053	10	0.7929	4	0.7797	0.6701
(5A, 5)	600	0.8571	15	0.8467	5	0.7365	0.5259
(5B, 5)	600	0.6849	15	0.6838	5	0.5599	0.4538
(5C, 5)	600	0.8068	15	0.7367	5	0.6652	0.5322
(5D, 5)	600	0.8602	15	0.7861	5	0.7680	0.6990

TABLE III

Evaluation of the greedy scheme's performance

Input	**Greedy vs. Exhaustive**	**Greedy vs. Random**	
(Obj, n)	$S(\mathbf{a}^\dagger)/S(\mathbf{a}^*)$	$\frac{S(\mathbf{a}^\dagger)-\mathbb{E}[S(\mathbf{a})]}{S(\mathbf{a}^\dagger)}$	$\frac{S(\mathbf{a}^\dagger)-S(\mathbf{a}_{\text{wc}})}{S(\mathbf{a}^\dagger)}$
(3A, 3)	98.22%	3.04%	7.07%
(3B, 3)	99.72%	2.72%	10.42%
(3C, 3)	99.91%	6.62%	11.71%
(3D, 3)	95.31%	1.24%	11.01%
(4A, 4)	98.09%	0.05%	1.10%
(4B, 4)	99.09%	8.92%	18.79%
(4C, 4)	99.05%	5.81%	27.96%
(4D, 4)	89.92%	1.67%	15.49%
(5A, 5)	98.79%	13.01%	37.89%
(5B, 5)	99.84%	18.12%	33.64%
(5C, 5)	91.32%	9.71%	27.76%
(5D, 5)	91.38%	2.30%	11.07%
Mean	96.72%	6.10%	17.83%

and unsafe. Thereafter, an ideal position for the cutter is prescribed and the decision's safety is measured. Safety is modeled to capture the decision's dynamic outcomes, *i.e.*, the structure's resulting kinetics, as well as the geometric outcomes, *i.e.*, the motion's proximity to the cutter's prescribed position. We design three decision search schemes to find a solution consisting of the decision sequence, its safety score, and the sequence of prescribed cutter positions. These schemes are tested in simulation against 12 partitioned objects wherein their safety performance and computational costs are measured. Our results show that the greedy decision search satisfices the problem with considerably less computational cost than exhaustive searching. In essence, our greedy algorithm: (1) scales better with input size n by computing $\mathcal{O}(n^2)$ rather than $\mathcal{O}(n!)$ decision evaluations; (2) yields good solutions (sometimes near-optimal); (3) is noticeably better than random selection (sometimes drastically); and, (4) is significantly better than the worst-case choice. In future work, it is worth incorporating more realistic dynamics and decision parameters to reduce the sim-to-real gap.

References

[1] W. R. Stahel, "The circular economy," *Nature*, vol. 531, no. 7595, pp. 435–438, 2016.

[2] M. Geissdoerfer, P. Savaget, N. M. Bocken, and E. J. Hultink, "The circular economy – a new sustainability paradigm?" *J. Clean. Prod.*, vol. 143, pp. 757–768, 2017.

[3] K. Jain and J. Pruijn, "An overview of the global ship recycling industry," in *Reference Module in Materials Science and Materials Engineering*, 2017, pp. 1–22.

[4] J. S. Ribeiro and J. de Oliveira Gomes, "Proposed framework for end-of-life aircraft recycling," *Procedia CIRP*, vol. 26, pp. 311–316, 2015, 12th Global Conference on Sustainable Manufacturing – Emerging Potentials.

[5] M. Sabaghi, Y. Cai, C. Mascle, and P. Baptiste, "Sustainability assessment of dismantling strategies for end-of-life aircraft recycling," *Resour. Conserv. Recycl.*, vol. 102, pp. 163–169, 2015.

[6] F. C. Rios, W. K. Chong, and D. Grau, "Design for disassembly and deconstruction - challenges and opportunities," *Procedia Engineering*, vol. 118, pp. 1296–1304, 2015, defining the future of sustainability and resilience in design, engineering and construction.

[7] N. A. W. A. Zawawi, M. S. Liew, and K. L. Na, "Decommissioning of offshore platform: A sustainable framework," in *IEEE Colloquium on Humanities, Science and Engineering*, 2012, pp. 26–31.

[8] H. Yan, L. Wu, and J. Yu, "The environmental impact analysis of hazardous materials and the development of green technology in the shipbreaking process," *Ocean Eng.*, vol. 161, pp. 187–194, 2018.

[9] J.-K. Choi, D. Kelley, S. Murphy, and D. Thangamani, "Economic and environmental perspectives of end-of-life ship management," *Resour. Conserv. Recycl.*, vol. 107, pp. 82–91, 2016.

[10] S. Lee, W. Pan, T. Linner, and T. Bock, "A framework for robot assisted deconstruction: Process, sub-systems and modelling," in *International Symposium on Automation and Robotics in Construction*, 2015.

[11] X. Wei, S. Qiu, L. Zhu, R. Feng, Y. Tian, J. Xi, and Y. Zheng, "Toward support-free 3D printing: A skeletal approach for partitioning models," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 10, pp. 2799–2812, 2018.

[12] L. Luo, I. Baran, S. Rusinkiewicz, and W. Matusik, "Chopper: Partitioning models into 3D-printable parts," *ACM Trans. Graph.*, vol. 31, no. 6, nov 2012.

[13] S. Christoph Stein, M. Schoeler, J. Papon, and F. Worgotter, "Object partitioning using local convexity," in *Proc. IEEE Conf. Comput.Vis. Pattern Recognit.*, June 2014.

[14] H. Poschmann, H. Brüggemann, and D. Goldmann, "Disassembly 4.0: A review on using robotics in disassembly tasks as a way of automation," *Chem. Ing. Tech.*, vol. 92, no. 4, pp. 341–359, 2020.

[15] F. Pistolesi and B. Lazzerini, "TeMA: A tensorial memetic algorithm for many-objective parallel disassembly sequence planning in product refurbishment," *IEEE Trans. Industr. Inform.*, vol. 15, no. 6, pp. 3743–3753, 2019.

[16] Y.-S. Ma, H.-B. Jun, H.-W. Kim, and D.-H. Lee, "Disassembly process planning algorithms for end-of-life product recovery and environmentally conscious disposal," *Int. J. Prod. Res.*, vol. 49, no. 23, pp. 7007–7027, 2011.

[17] H.-J. Han, J.-M. Yu, and D.-H. Lee, "Mathematical model and solution algorithms for selective disassembly sequencing with multiple target components and sequence-dependent setups," *Int. J. Prod. Res.*, vol. 51, no. 16, pp. 4997–5010, 2013.

[18] M.-L. Lee, S. Behdad, X. Liang, and M. Zheng, "Disassembly sequence planning considering human-robot collaboration," in *Proc. Am. Control Conf.*, 2020, pp. 2438–2443.

[19] W. Xu, Q. Tang, J. Liu, Z. Liu, Z. Zhou, and D. T. Pham, "Disassembly sequence planning using discrete bees algorithm for human-robot collaboration in remanufacturing," *Robot. Comput. Integr. Manuf.*, vol. 62, p. 101860, 2020.

[20] Y. Laili, X. Li, Y. Wang, L. Ren, and X. Wang, "Robotic disassembly sequence planning with backup actions," *IEEE Trans. Autom.*, vol. 19, no. 3, pp. 2095–2107, 2022.

[21] M. Kheder, M. Trigui, and N. Aifaoui, "Optimization of disassembly sequence planning for preventive maintenance," *Int. J. Adv. Manuf. Technol.*, vol. 90, no. 5, pp. 1337–1349, 2017.

[22] F. Giudice and G. Fargione, "Disassembly planning of mechanical systems for service and recovery: a genetic algorithms based approach," *J. Intell. Manuf.*, vol. 18, no. 3, pp. 313–329, Jun. 2007.

[23] K.-K. Seo, J.-H. Park, and D.-S. Jang, "Optimal disassembly sequence using genetic algorithms considering economic and environmental aspects," *Int. J. Adv. Manuf. Technol.*, vol. 18, no. 5, pp. 371–380, 2001.

[24] L. Galantucci, G. Percoco, and R. Spina, "Assembly and disassembly planning by using fuzzy logic & genetic algorithms," *Int. J. Adv. Robot. Syst.*, vol. 1, no. 2, p. 7, 2004.

[25] S. Vongbunyong, S. Kara, and M. Pagnucco, "A framework for using cognitive robotics in disassembly automation," in *Leveraging Technology for a Sustainable World*, 2012, pp. 173–178.

[26] X. Guo, S. Liu, M. Zhou, and G. Tian, "Disassembly sequence optimization for large-scale products with multiresource constraints using scatter search and petri nets," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2435–2446, 2016.

[27] Y. Tang, M. Zhou, and R. Caudill, "An integrated approach to disassembly planning and demanufacturing operation," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 773–784, 2001.

[28] Y. Tang, "Learning-based disassembly process planner for uncertainty management," *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 39, no. 1, pp. 134–143, 2009.

[29] C. He, Z. Jin, R. Gu, and H. Qu, "Automatic disassembly and recovery device for mobile phone circuit board cpu based on machine vision," *J. Phys. Conf. Ser.*, vol. 1684, no. 1, p. 012137, nov 2020.

[30] A. ElSayed, E. Kongar, S. M. Gupta, and T. Sobh, "A robotic-driven disassembly sequence generator for end-of-life electronic products," *J. Intell. Robot. Syst.*, vol. 68, no. 1, pp. 43–52, 2012.

[31] M. Merdan, W. Lepuschitz, T. Meurer, and M. Vincze, "Towards ontology-based automated disassembly systems," in *Annual Conference on IEEE Industrial Electronics Society*, 2010, pp. 1392–1397.

[32] A. S. Allam and M. Nik-Bakht, "From demolition to deconstruction of the built environment: A synthesis of the literature," *J. Build. Eng.*, vol. 64, p. 105679, 2023.

[33] M. Helal and T. Bock, "Towards a sustainable construction industry: Automated deconstruction and recycling," *Democratic Transition and Sustainable Communities*, p. 611, 2013.

[34] B. Sanchez, C. Rausch, C. Haas, and R. Saari, "Multi-objective optimization analysis for selective disassembly planning of buildings," in *International Symposium on Automation and Robotics in Construction*, Banff, Canada, May 2019, pp. 128–135.

[35] E. Queheille, F. Taillandier, and N. Saiyouri, "Optimization of strategy planning for building deconstruction," *Autom. Constr.*, vol. 98, pp. 236–247, 2019.

[36] C. Kasperzyk, M.-K. Kim, and I. Brilakis, "Automated re-prefabrication system for buildings using robotics," *Autom. Constr.*, vol. 83, pp. 184–195, 2017.

[37] E. Lublasser, L. Hildebrand, A. Vollpracht, and S. Brell-Cokcan, "Robot assisted deconstruction of multi-layered façade constructions on the example of external thermal insulation composite systems," *Construction Robotics*, vol. 1, no. 1, pp. 39–47, 2017.

[38] E. Lublasser, K. Iturralde, T. Linner, S. Brell Cokcan, and T. Bock, "Automated refurbishment & end-of-life processes research approaches in German and Japanese construction." in *CIB IAARC W119 CIC Workshop*, 2016.

[39] B. Sanchez, C. Rausch, C. Haas, and T. Hartmann, "A framework for BIM-based disassembly models to support reuse of building components," *Resour. Conserv. Recycl.*, vol. 175, p. 105825, 2021.

[40] B. Sanchez, C. Rausch, and C. Haas, "Deconstruction programming for adaptive reuse of buildings," *Autom. Constr.*, vol. 107, p. 102921, 2019.

[41] B. Sanchez, C. Rausch, C. Haas, and T. Hartmann, "VPL prototypes for determining disassembly parameters of BIM models," in *International Conference on Civil Engineering and Architecture*, Singapore, 2022, pp. 507–514.

[42] C. Rausch, B. Sanchez, and C. Haas, "Spatial parameterization of non-semantic cad elements for supporting automated disassembly planning," *Modular and Offsite Construction Summit Proceedings*, pp. 108–115, 2019.

[43] R. Volk, T. H. Luu, J. S. Mueller-Roemer, N. Sevilmis, and F. Schultmann, "Deconstruction project planning of existing buildings based on automated acquisition and reconstruction of building information," *Autom. Constr.*, vol. 91, pp. 226–245, 2018.

[44] X. J. Ge, P. Livesey, J. Wang, S. Huang, X. He, and C. Zhang, "Deconstruction waste management through 3D reconstruction and BIM: a case study," *Vis. Eng.*, vol. 5, no. 1, p. 13, 2017.

[45] Y. Wei, A. Pushkar, and B. Akinci, "Supporting deconstruction waste management through 3D imaging: A case study," in *International Symposium on Automation and Robotics in Construction*, May 2019, pp. 438–445.

[46] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE Int. Conf. Intell. Robots Syst.*, vol. 3, 2004, pp. 2149–2154 vol.3.