

Feature-driven Next View Planning for Cutting Path Generation in Robotic Metal Scrap Recycling

James Akl, *Graduate Student Member, IEEE*, Fadi Alladkani, *Graduate Student Member, IEEE*,
and Berk Calli, *Member, IEEE*

Abstract—Metal recycling in scrapyards, where workers cut decommissioned structures using gas torches, is labor-intensive, difficult, and dangerous. As global metal scrap recycling demands are rising, robotics and automation technologies could play a significant role to address this demand. However, the unstructured nature of the scrap cutting problem—due to highly variable object shapes and environments—poses significant challenges to integrate robotic solutions. We propose a novel collaborative workflow for robotic metal cutting that combines worker expertise with robot autonomy. In this workflow, the skilled worker studies the scene, determines an appropriate cutting reference, and marks it on the object with spray paint. The robot, then, autonomously explores the surface of the object for identifying and reconstructing the drawn reference, converts it to a cutting trajectory, and finally executes the cut. This paper focuses on the surface exploration and cutting reference reconstruction tasks, which require appropriate next view planning (NVP) algorithms. We devise three NVP algorithms enabling the robot to explore and extract desired features from the scene, *i.e.*, the drawn reference, without requiring any *a priori* object model. Contrasting with global or feature-agnostic NVP algorithms, our approaches guide the robot via desired local features to increase the efficiency of the exploration. We evaluate our NVP algorithms against six categories of objects both in simulation and in physical experiments.

Note to Practitioners—This work is motivated by the need of extracting a desired cutting reference determined and drawn on the object by scrap yard workers. From the robot's perspective, it must explore and reconstruct the drawing, starting from an unknown scene containing an unknown object featuring an unknown drawing. We assume that an RGB-D camera is attached to the tool-tip of the robot, and the color of the drawn path is significantly different from the object's color. The goal of the robotic system is to explore the object surface to uncover the drawn path entirely without colliding with the object. The exploration algorithm must overcome complex object shapes and must be fast enough for practical use in scrap yards. This means conventional exploration (active vision) techniques are insufficient since they focus on exploring the entirety of the object, which is unnecessary for our task and is time-consuming. Our methods exploit the drawing information to guide the exploration for quickly determining a suitable viewpoint, which results in an efficient extraction of the entire cutting reference, without needing to explore the entire object's surface. Our algorithms are robust against adversarial features such as discontinuous, non-smooth, or self-occluded object surfaces. Our feature-driven strategies are not limited to robotic scrap cutting as they are applicable to any viewpoint planning problem requiring high performance while extracting the local features in the scene.

Index Terms—Active vision, human-robot collaboration, robotic cutting, metal recycling, 3-D feature reconstruction.

The authors are with the Robotics Engineering Department, Worcester Polytechnic Institute, 27 Boynton St, Worcester, MA 01609, USA.
E-mail: {jgakl, fmalladkani, bcalli}@wpi.edu

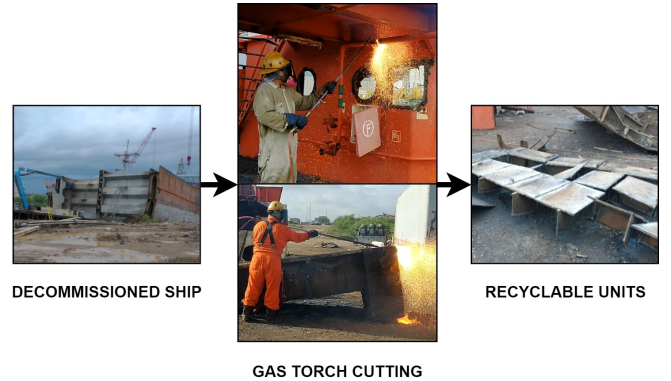


Fig. 1. Shipbreaking operations break down a decommissioned vessel into salvageable parts, further cutting them into recyclable units.

I. INTRODUCTION

SHIPPING of goods by sea plays a vital role in the global economy. Transport vessels typically have 25–30 years of service [1], at the end of which, they are sent to shipbreaking yards for recycling. During this scrapping process, the vessels are broken apart and then cut into recyclable metal components using gas torches (see Fig. 1). The working conditions in these shipbreaking yards are difficult and dangerous, often requiring the disposal of hazardous substances [2]. This industry is characterized by its labor-intensive and comparatively low-technology activity—shipbreaking operations are not straightforward to mechanize or automate.

Recent years observed increases in the decommissioning of broken, aging, or obsolete vessels. However, shipbreaking capacity is limited [3], especially in industrialized countries [2] due to higher labor costs. To address these rising global scrapping demands, the recycling capacity of scrapyards needs to be increased [4]. Robotics and automation have a potential to enable higher throughput in shipbreaking operations and help address the growing needs of the metal scrap industry, while also improving the safety of scrapyard workers.

However, traditional automation approaches are not well-suited for the diverse operations and chaotic environments of shipbreaking yards. Specifically, automating metal cutting in the scrapyard is difficult because of three salient challenges:

- 1) The inputs—*i.e.*, the scrap pieces to be cut—have highly-variable properties such as size, shape, material type, surface condition, and spatial configuration.
- 2) The torch cutting operation itself has parameters such as cutting speed, approach distance and angle, tip settings, and oxy-fuel flow proportions, all of which depend on

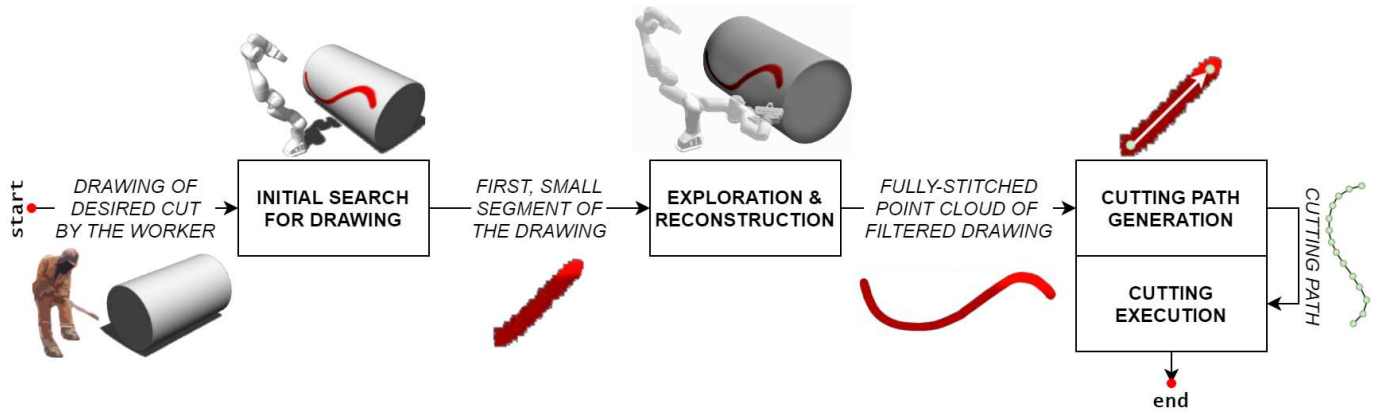


Fig. 2. Block representation of the system's general workflow. This paper's primary contribution concerns the "Exploration & Reconstruction" component.

the aforementioned input properties and are difficult to determine.

- 3) The scrapyards are unstructured and have challenging workflows and conditions.

We address the first and third challenges in determining a desired cutting path on the target object; we do not address the second challenge of oxy-fuel cutting control. One outcome of this work is to provide a reference cutting path to such a controller for oxy-fuel cutting. Thus, for each piece's individual properties and surrounding conditions, the desired cutting locations must be determined, the subsequent cutting trajectories must be generated, and the resultant cut must be executed with the adequate cutting parameters. In manual cutting, skilled workers can successfully determine the suitable operation objectives, and can apply the adequate cutting parameters—all while compensating for disturbances. Translating all of these decision-making capacities and workers' implicit know-how into robot task parameters is highly challenging. While there are a few studies in the literature on robotic cutting [5]–[10], these papers focus on structured conditions, restricted tasks, controlled environments, or known object shapes. The development of robotic cutting systems that address the needs of the metal scrap cutting industry is not examined in the literature.

A. Overview of the Proposed Scheme

We propose a novel human-robot collaboration scheme that combines the respective strengths of robots and skilled workers. In brief, the skilled worker inspects the target piece to determine an adequate cut. They then draw the cutting path on the object surface using an adequate marker, *e.g.*, spray paint. Next, the vision-equipped robot examines the target object to locate the drawn path. The robot explores the drawing along the object to fully reconstruct it. With the acquired path and object properties, cutting trajectories are generated and the robot executes the cut.

This collaboration yields pertinent advantages:

- We leverage worker expertise that is hard to transfer to the robot. The worker provides cutting references to the robot by conveniently drawing them on an object's surface.
- The work required by the robot is significantly reduced. The robot explores only the marked regions on the object—this avoids wasteful scanning of the entire object.
- The worker's job is greatly simplified. The tedious and high-risk tasks are delegated to the robot.

The proposed scheme's components are shown in Fig. 2.

B. Paper's Focus and Contribution

The scheme in Fig. 2 has two main tasks for the robot: extract the cutting reference and execute the cut. This paper focuses on exploring the target object's surface and extracting the 3-D drawing to generate a desired cutting path.

It is important to note that the objects and drawings are both unknown to the robot prior to the operation, as they would be in scrapyards. Moreover, the objects in scrapyards are often large and complex in shape, which requires a systematic exploration of the objects' surface; the drawings would not be visible from a single viewpoint of the camera and are often times self-occluded. Therefore, the robot needs to scan the drawing one image at a time and reconstruct it piece by piece on a surface whose size and shape are both unknown. For this, the robot must carefully and intelligently plan its camera's next poses (viewpoints). This goal requires next view planning (NVP), *i.e.*, determining the camera's subsequent poses to fully explore the object's targeted feature. This paper builds on our prior work [11], which outlined our human-robot collaboration scheme and presented our first NVP algorithm for cutting path reconstruction. While this algorithm was effective for objects with simpler geometries, it failed against more complex shapes by terminating prematurely. In this work, we address our prior method's shortcomings by adopting a next-best view strategy repurposed from the information gain formulations [12], [13].

This paper accomplishes the following contributions:

- We propose a novel feature-driven active vision strategy to autonomously search for and reconstruct a drawn cutting reference on an object's surface, both having unknown size and shape.
- We develop two next best view (NBV) algorithms which respectively constrain and guide the search using feature information to select the camera's next viewpoint.

- We provide simulations and real robot experiments with six different object categories to evaluate the efficiency, flexibility, and robustness of our methods.

To the best of our knowledge, this work is the first to develop solutions for robotic cutting in unstructured scrapyards; and the first to present feature-driven active vision schemes for efficient and robust surface exploration. Our novel algorithms are not limited to scrap cutting but can also be applied to other problems that require 3-D feature reconstruction.

II. RELATED WORK

This section, first, situates existing methodologies for robotic cutting and welding against our problem's challenges. Next, we discuss how our approach compares to current standards in active vision and NVP.

A. Cutting Path Generation for Robotic Cutting

As reviewed in [5], robotic cutting systems are employed in various industrial and medical applications. While these robots are highly diverse in their designs, objectives, and cutting media (e.g., laser, plasma, oxy-gas, or water-jet), they often benefit from executing explicit and well-defined tasks in structured and controlled environments. For example, in manufacturing applications, object properties are known and can be directly modeled for the robot, as is the case for offline robot programming in [6] and [7]. Similarly, [8] improves the robot's cutting efficiency on known airframe parts in known locations to reduce manufacturing costs. For automated laser cutting, analytical methods in [14]–[18] are developed for known object geometries, while [19] improves path generation in a known and structured scene. In gas cutting, [10] develops a reactive control architecture for a robot to remove low-quality metal strips from sheet metal moved on a conveyor. Each of these uses contrasts with the operations of metal recycling scrapyards, where objects, their surroundings, and their cutting plans are unknown *a priori*. To the best of our knowledge, the only work on robotic shipbreaking is [9], where the focus is on applying a specialized cutting medium (hybrid induction plasma) to submarine recycling. This system operates only on submarine hulls, and therefore cannot handle the more difficult cutting operations in scrapyards. These formulations for robotic cutting are incompatible with our target application—metal scrap recycling—since they operate in structured environmental settings and directly on objects with full or partial knowledge of their properties. Instead, our algorithms explore and reconstruct an unknown drawing (cutting reference) on unknown objects. Our algorithms operate in different problem conditions, and so cannot be meaningfully compared against the aforementioned methods. We are not aware of any existing work on robotic cutting in unstructured environments—except ours in [11], which this paper extends.

B. Weld Seam Extraction for Robotic Welding

While welding is a distinct operation from cutting, there are similar subtasks required for its automation. For example, the problem of weld seam extraction, which is covered in

[20]–[26] has apparent similarities with retrieving a desired cutting path. Elsewhere, [27] proposes human-robot collaborative welding using a virtual reality setting. However, for scrap cutting, this approach limits the robot's task autonomy. It is worth noting that in welding, the goal is to join materials for the purpose of fabrication wherein there is abundant and reliable knowledge about the input objects. In effect, some of these seam extraction methods either assume partial or full knowledge of the seam, or expect a viewpoint containing it entirely. By contrast, the goal in metal recycling is to scrap or break down the objects into smaller workable units; noting that there is a large variety of objects and a high variance in their properties. Moreover, weld seams have favorable properties (precise, clear, and relatively predictable) which are often exploited to facilitate the seam extraction. In metal scrap cutting, however, the cutting references are noisy, irregular, and unpredictable. This is worsened by the fact that these drawings are painted by the worker in an unstructured setting. Moreover, scrapyard objects are often too large and complex to guarantee that the entire drawing can be fit in one view. For these reasons, these weld seam extraction methods are incompatible with our problem nor can they be directly compared with our algorithms.

C. Active Vision and Next View Planning

Active vision systems investigate the environment to gain pertinent information by manipulating the camera's viewpoint. Viewpoint planning has diverse applications, among which include precise manufacturing [28], shape reconstruction [29], human motion capture [30], underwater exploration [31], and robot grasping [32]–[34]. Recent developments also tackle less conventional configurations and solutions, such as exploring articulated scenes (manipulating surroundings while scanning) [35], scanning objects and scenes using an aerial robot [36], and estimating occlusions via surface edge exploration [37]. Learning-oriented paradigms are also found in NVP, most notably deep learning [38]–[40] and reinforcement learning [41]. These approaches are not suitable for our application since they do not exploit the relevant feature information to enhance the NVP. Instead, we avoid wasteful scanning of the entire object and target only its desired subsets.

Two of the algorithms proposed in this paper are probabilistic NBV planners and use formulations similar to the information gain-based methods found in [12] and [13]. In these works, a probabilistic volumetric map is used for volumetric scene reconstruction, where the expected information gain is computed from discrete candidate views. We emphasize that our NBV algorithms repurpose the aforementioned methods in a specialized framework adapted to the domain-specific task of robotic metal scrap scanning and cutting. While the formulations [12] and [13] are useful for global scene exploration and total object reconstruction, they are not tailored for nor are incompatible with our target application since they seek to map an entire object or scene and not the subset of the object surface containing the desired feature—thereby solving a distinct problem. In some cases, it may not even be possible—in our problem's formulation—to scan the object

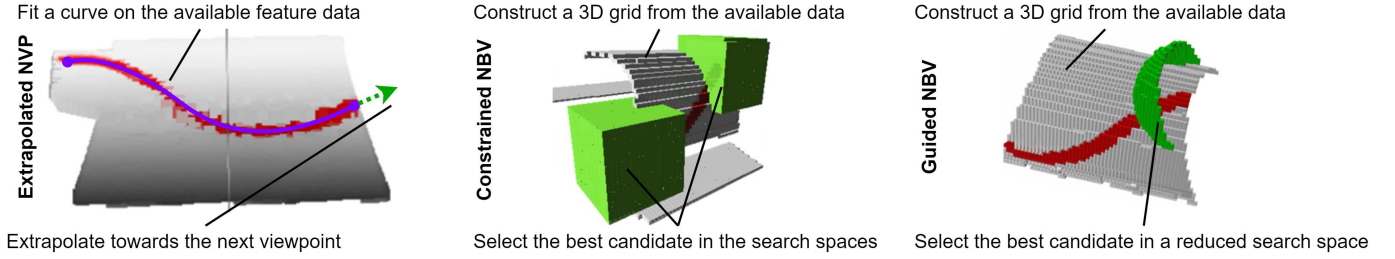


Fig. 3. Visual overview of the three proposed next view planning algorithm. “Extrapolated NVP” fits a curve to the feature points in the available data and obtains the next viewpoint pose by extrapolating from this curve while estimating surface normals. “Constrained NBV” converts the available data into a volumetric occupancy grid, generates search spaces, scores every candidate viewpoint within, and then selects the candidate with the highest score. “Guided NBV” constructs the same grid but instead generates a reduced search space thereby rapidly obtaining a next viewpoint.

entirely due to workspace constraints within which our methods operate. Accordingly, we adapt their strategies into feature-driven planners by assisting the viewpoint search using the desired feature’s information. To the best of our knowledge, no existing approach targets local features on unknown objects in unstructured scenes, a need which we address in this paper.

III. OVERVIEW OF THE SYSTEM

This section presents the different workflow components, the robot tasks, and the hardware configuration for a contained and structural understanding of the project.

A. Hardware Configuration

We design our algorithms assuming that the system includes a mobile platform, a manipulator, an RGB-D camera, and a gas torch connected to appropriate oxyfuel tanks (see Fig. 4). Note that for scrap cutting, the fuel of choice is commonly oxypropane. At the current stage—both in simulation and in the physical experiments—the manipulator in use is the Franka Emika Panda 7-DOF arm and the RGB-D camera equipped is the Intel RealSense D435. Currently, the Panda robot is used as stand-alone, and its integration to a mobile platform will be addressed in our future work.

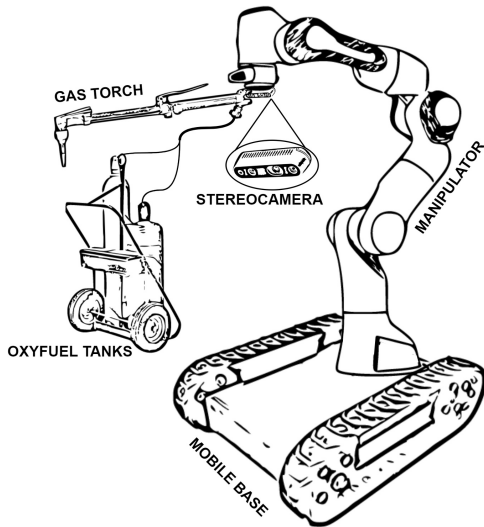


Fig. 4. Conceptual diagram of the system’s hardware configuration.

B. Workflow Components

In this collaborative workflow, the worker and robot have well-defined roles and functions. From the robot’s perspective, the worker provides the cutting references on the object. Afterwards, the robot must discover and recover these references and then carry them out. We emphasize that the robot does not know the locations of neither drawing nor object, nor does it assume their shape or size. It merely assumes their existence, and must autonomously discover and acquire any required properties. This is essential for scrapyard environments, as there is little regularity in shapes, sizes, and cuts.

After worker input, the robot’s tasks become: (see Fig. 2)

- 1) Search to discover an initial partial view of the drawing.
- 2) Plan next views to gradually reconstruct the full drawing.
- 3) Generate a 3-D cutting path from this recovered drawing.
- 4) Generate a cutting trajectory within known constraints.
- 5) Execute the cut and update it with RGB-D feedback.

The workflow can be viewed as an action sequence on the input (colored point clouds) to the desired output (successful cut). This is: scan (tasks #1 and #2), generate the trajectory (tasks #3 and #4), and execute with feedback (task #5).

The main contributions of this paper solve tasks #2 (obtaining the drawing) and #3 (generating a cutting path). For this purpose, task #1 is assumed complete, *i.e.*, we assume that a (small) part of the drawing is visible to the robot from the initial view.

IV. EXPLORATION & RECONSTRUCTION

This section introduces and explains the subroutines of the exploration and reconstruction task.

A. Task Overview

Referring to Fig. 2, the exploration and reconstruction task must facilitate the eventual generation of a cutting path. Instrumentally, it must provide an appropriate reconstruction of the desired feature (here, a 3-D colored drawing). This reconstructed drawing must cover as much of the original drawing as possible. Thus, the task’s desired output is a high-coverage point cloud reconstruction of the drawing.

Due to the high variance of object sizes and shapes, the robot’s initial view is unlikely to contain the entire drawing. Accordingly, the robot keeps exploring the object until it fully uncovers the desired feature. Thus, the exploration and

reconstruction routine is formulated as a loop that terminates once the drawing is considered fully explored. Until this stop condition is met, the robot repeats the following steps:

- 1) Acquire a colored point cloud image.
- 2) Transform the acquired cloud to the fixed base frame.
- 3) Combine clouds from several views to reconstruct a surface containing parts of the drawing.
- 4) Segment the drawing region from the reconstruction.
- 5) Plan the next viewpoint using the extracted points.
- 6) Move to the next view while avoiding collisions.

In essence, we iteratively reconstruct a subset of the object's surface that contains the entire drawing, and then at each step, we use its available portion of the drawing to inform the next viewpoint. We terminate the search once this reconstruction is considered to contain the entire drawing, which is then extracted. These steps are elaborated in the rest of this section.

B. Point Cloud Processing

The robot acquires colored point clouds obtained from the variable camera frame. The surface containing the drawing is reconstructed by combining the point clouds obtained from multiple views. These clouds are first transformed to the fixed base frame and are then concatenated.

More formally, let $k \geq 0$ be the exploration's current step. Let cC_k be the cloud obtained at step k with respect to the camera frame c . The camera's pose at step k is bT_k in the base frame b . These acquired clouds cC_k are concatenated to iteratively expand the cumulative knowledge of the drawing.

Accordingly, let ${}^bC_{\text{total},k}$ be the cloud containing all the RGB-D information obtained thus far, *i.e.*, the cumulative cloud after step k expressed in the base frame. This can be expressed recursively for $k \geq 1$ as follows,

$$\begin{cases} {}^bC_{\text{total},0} = {}^bT_0 {}^cC_0, \\ {}^bC_{\text{total},k} = {}^bC_{\text{total},k-1} \cup {}^bT_k {}^cC_k \end{cases} \quad (1)$$

The transformations bT_k map all clouds acquired at different steps k to the base frame for concatenation. The expression ${}^bT_k {}^cC_k$ maps all of the cloud's member points from the camera frame at step k to the base frame.

The loop's next step is to segment the drawing from the current cumulative cloud ${}^bC_{\text{total},k}$. Let $\phi_\xi(\cdot)$ be the filtering function defined by its parameter ξ which determines its filtering behavior. In our setup, $\phi_\xi(\cdot)$ filters by color such that ξ is an admissible range of colors. Let ${}^bD_{\text{total},k}$ be the cloud representing the drawing such that $\phi_\xi : {}^bC_{\text{total},k} \mapsto {}^bD_{\text{total},k}$.

With these first four subtasks defined, we summarize their inputs, outputs, and interaction in the pseudocode below.

```
define ProcessClouds(step  $k$ ):
   ${}^cC_k \leftarrow \text{AcquireImageAt}({}^bT_k)$ 
   ${}^bC_k \leftarrow \text{Transform}({}^bT_k, {}^cC_k)$ 
   ${}^bC_{\text{total},k} \leftarrow \text{Concatenate}({}^bC_{\text{total},k-1}, {}^bC_k)$ 
   ${}^bD_{\text{total},k} \leftarrow \text{Filter}(\xi, {}^bC_{\text{total},k})$ 
  return ( ${}^bD_{\text{total},k}, {}^bC_{\text{total},k}, {}^bC_k$ )
```

We implement some of these 3-D point cloud processing tasks using the Point Cloud Library [42].

We note the distinction between the clouds ${}^bC_{\text{total},k}$ and ${}^bD_{\text{total},k}$. The cloud ${}^bC_{\text{total},k}$ is the local surface reconstruction of the object, *i.e.*, the concatenation of all the acquired and transformed point clouds bC_k at each step k . In contrast, the cumulative feature cloud ${}^bD_{\text{total},k}$ is the filtered version of ${}^bC_{\text{total},k}$ and thus retains only the feature points (in our case, points of red color) with all other points discarded.

C. Next View Planning

Until now, we have defined the acquisition and processing of point clouds at a particular viewpoint pose bT_k . The initial viewpoint at $k = 0$ supposedly provided by the mobile search (see Fig. 2) is assumed given. Beyond this, we must obtain the subsequent viewpoints to gradually explore and reconstruct the drawing. For this, we require a next view planner.

Conceptually, the NVP algorithm performs higher-level reasoning on the raw point clouds obtained from the ProcessClouds($\cdot$) procedure. Specifically, the NVP generates candidate viewpoints using information from the cumulative feature cloud ${}^bD_{\text{total},k}$, the local surface cloud ${}^bC_{\text{total},k}$, and the latest transformed cloud bC_k . The pseudocode below conceptually sketches the exploration and reconstruction routine.

```
define ReconstructDrawing():
   ${}^bT_0 \leftarrow \text{InitialViewpoint}()$ 
  for  $k = 0, 1, \dots, k_{\text{stop}} - 1$ :
    ( ${}^bD_{\text{total},k}, {}^bC_{\text{total},k}, {}^bC_k$ )  $\leftarrow \text{ProcessClouds}(k)$ 
     ${}^bT_{k+1} \leftarrow \text{NextViewpoint}({}^bD_{\text{total},k}, {}^bC_{\text{total},k}, {}^bC_k)$ 
   ${}^bD_{\text{total},k_{\text{stop}}} \leftarrow \text{ProcessClouds}(k_{\text{stop}})$ 
  return  ${}^bD_{\text{total},k_{\text{stop}}}$ 
```

The loop's stopping condition is determined and checked by the viewpoint planner, terminating the exploration at some eventual step k_{stop} . After termination, the cumulative cloud ${}^bC_{\text{total},k_{\text{stop}}}$ should contain the entirety of the desired feature. The final and fully reconstructed drawing's point cloud ${}^bD_{\text{total},k_{\text{stop}}}$ is outputted for cutting trajectory generation.

Each of our three planning algorithms implement the ReconstructDrawing() routine with their own respective mechanism for NextViewpoint($\cdot$), *i.e.*, planning the next viewpoint. Irrespective of choice, the viewpoint planner controls these two decisions during exploration:

- 1) *Termination*: Determine if the drawing is fully explored, and accordingly either proceed searching or terminate.
- 2) *Viewpoint Generation*: Provide and select candidate camera poses to continue the robot's search.

For scrap cutting, the NVP algorithm is subject to performance constraints. An inefficient planner slows down the exploration routine, which would worsen cutting productivity. The planning time is affected jointly by the number of steps and by the step duration. This often comes with a trade-off, as planners that finish with less steps tend to spend more time per view, and by contrast planners which compute steps rapidly tend to iterate more. This trade-off is present in our algorithms and is later examined in our evaluations.

The NVP algorithm determines poses to visit sequentially, throughout the exploration task, to reconstruct the drawing on the object surface. The motion planner attempts to plan

a feasible trajectory towards these poses and executes the first viable one. This motion then executes while avoiding collisions. For our implementation, we use ROS [43] and Gazebo [44]. We also use MoveIt! [45] for motion planning, which is internally set to use OMPL and TRAC-IK.

V. NEXT VIEW PLANNING ALGORITHMS

This section develops the three feature-driven NVP algorithms. Each algorithm:

- Obtains a point clouds of the object that is marked with the cutting path.
- Segments the points that belong to the cutting path.
- Processes the object point cloud and the cutting path point cloud to calculate the next viewpoint.
- Moves the robot to the next viewpoint, and repeats this process until a termination condition is reached.

The algorithms can be distinguished (see Fig. 3) by how each of these three high-level actions is accomplished.

The first algorithm generalizes the procedure for point cloud fitting and curve extrapolation, from our prior work [11]. To the best of the authors' knowledge, this method is the first of its kind to address the application of interest: reconstructing an unknown drawing on an unknown surface, given a known feature (here, the color red). As such, this algorithm is used as a baseline against which the remaining two are compared.

The second algorithm reformulates the methodologies in [12] and in [13] into one that is adapted for the application of interest. Briefly, the method relies on a probabilistic occupancy voxel grid and a quality metric (based on information gain) to explore and rank candidate viewpoints, thereby selecting that which maximizes quality. This planner searches the grid in a rather exhaustive manner. In contrast, our third algorithm, which uses the same formulation as the second, exploits greedy-like optimizations for faster searching.

We examine each method in its respective subsection.

A. Extrapolated Next View Planning (E-NVP) Algorithm

This first planner treats the feature NVP task like a path exploration problem—by iteratively exploring the branches of an unknown path until all endpoints are found. The point clouds are converted to more useful and more structured representations using fitting methods. The next view is obtained by extrapolating those fits.

Exploring along a path requires a sense of its direction. However, the data obtained from the stereocamera, which is then processed in the procedure `ProcessClouds(·)`, remains in the format of point clouds. Essentially, this is an unordered list of 3-D colored points where the direction of the drawing cannot be directly inferred. In this form, viewpoint planning on the desired feature is not straightforward.

This planner solves this representation problem by mapping the point cloud to a more usable structure. The cloud data is used to construct or fit spatial curves parametrized in 1-D. This procedure is captured by the map $\mathbb{R}_{xyzRGB}^6 \rightarrow \mathbb{R} \times \mathbb{R}_{xyz}^3$ where the input space $\mathbb{R}_{xyzRGB}^6$ represents the point cloud. The output space (a curve) can equivalently be rewritten as a rule

$\mathbb{R} \rightarrow \mathbb{R}_{xyz}^3$, which takes a 1-D parameter and returns a 3-D point. The cloud is thus reduced to a curve as follows,

$$\left\{ \left({}^b p_x, {}^b p_y, {}^b p_z, {}^b p_R, {}^b p_G, {}^b p_B \right)_j \right\}_{j=1} \xrightarrow{\text{Fit}} \left({}^b \mathbf{q}_{xyz}(t) \right)_t \quad (2)$$

This 1-D ordering of 3-D points yields a sense of direction along the curve wherein there is an ordering on the curve points. Specifically, the set of unordered points $\{{}^b \mathbf{p}_j\}_{j=1}$ is used to create the list of ordered points $\left({}^b \mathbf{q}_{xyz}(t) \right)_t$ indexed by the parameter t . This provides the planner with a way to determine the next viewpoint to continue exploring the drawing. Since the curve represents one continuous and known portion of the drawing, the drawing's unknown regions come after the endpoint of this curve. This planner assumes that the entire drawing has two extremities, meaning it is branchless. Thus, after the initial viewpoint, there are two scenarios for curve endpoint selection. If the initial viewpoint happened to already contain one extremity of the drawing, then the planner explores in the direction of the second curve endpoint. On the other hand, if the initial view contains an intermediate portion of the drawing, then the planner has two candidate directions to explore. The curve endpoint closest to the camera is chosen to reduce movement time between unexplored endpoints.

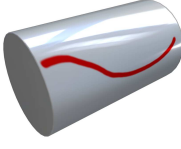
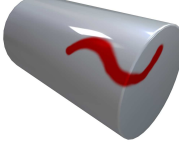
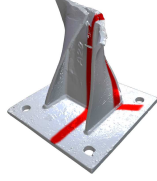
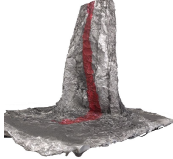
With a chosen curve endpoint, the planner extrapolates to the next viewpoint at each step at a configurable distance δ_{distance} . When $\delta_{\text{distance}} = 0$, the extrapolated endpoint itself becomes the next viewpoint. Exploring one curve endpoint at a time is slower, but plans towards the drawing extremity more conservatively. Alternatively, $\delta_{\text{distance}} > 0$ places the next viewpoint, beyond the curve endpoint. This speeds up exploration but may cause some overshoot. Accordingly, keeping the extrapolation distance small is preferable.

Extrapolation only determines the position ${}^b \mathbf{d}_{k+1}$ of the next viewpoint. To fully provide the next pose ${}^b \mathbf{T}_{k+1}$, the planner must also determine the next orientation ${}^b \mathbf{R}_{k+1}$ of the camera at that position ${}^b \mathbf{d}_{k+1}$. The planner sets the orientation to be orthogonal to the surface at the viewpoint position ${}^b \mathbf{d}_{k+1}$. This orientation is obtained by estimating the vector normal to the surface surrounding the viewpoint position ${}^b \mathbf{d}_{k+1}$. This region is contained in the cumulative cloud ${}^b C_{\text{total},k}$. Orienting the camera orthogonally helps the robot obtain a more accurate capture of the surface and thus avoid leaving gaps in the drawing point cloud during exploration—which would otherwise require backtracking and cost additional time. The orientation about the normal does not affect output significantly, and is relaxed for motion planning.

This planner continues searching by iteratively obtaining new clouds, fitting a curve on the desired feature, and extrapolating towards the next viewpoint. The planner considers a drawing extremity to be found when the size difference between two consecutive reconstructions ${}^b D_{\text{total},k}$ falls below a threshold δ_{size} . After both extremities are found, the search terminates and returns the reconstruction ${}^b D_{\text{total},k_{\text{stop}}}$. This iterative extrapolation procedure is outlined in Algorithm 1, where the result is the feature's reconstruction.

Note that this procedure is affected by the fitting method choice. In our prior work [11], we examined and compared

TABLE I
FEATURE CATEGORIES AND TEST OBJECTS FOR THE SIMULATION AND THE PHYSICAL EXPERIMENTS

Features	Category 1 Smooth Cylinder Smooth surface	Category 2 Sharp Cylinder Sharp occlusion	Category 3 Round Tank Smooth occlusion	Category 4 T-Piece Sharp transition	Category 5 I-Beam Nonsmooth surface	Category 6 Basket Discontinuous surface
Simulation Objects						
Experiment Objects						

the effects and results of two types of fitting methods. The first fitting method relies on an optimization scheme to fit a non-uniform rational B-spline (NURBS) curve to the cloud. This method initializes a proposed curve and then improves the fit using point-distance minimization. An alternative fitting method from computational geometry is topological skeletonization. The goal of a skeletonization algorithm is to compute the medial axis of a shape—defined as the set of points equidistant from its boundary. This discretizes the shape into a voxel grid, and then erodes its edges away by repeatedly applying a thinning algorithm, until a single voxel-wide skeleton remains. This method runs faster than the NURBS-based method, and is used in our evaluations.

Algorithm 1: Extrapolated Next View Planner (E-NVP)

Input : Initial view containing part of the drawing.
Output: ${}^bD_{\text{total},k_{\text{stop}}}$, fully-reconstructed drawing cloud.
Initialize step, extremities found, and viewpoint.
 $\text{step } k \leftarrow 0$
 $n_{\text{extremities}} \leftarrow 0$
 ${}^bT_0 \leftarrow \text{InitialViewpoint}()$
Explore the drawing until two extremities are found.
while $n_{\text{extremities}} < 2$ **do**
 Acquire, transform, concatenate, and filter clouds.
 $({}^bD_{\text{total},k}, {}^bC_k) \leftarrow \text{ProcessClouds}(k)$
 Backtrack to initial view if drawing size unchanged.
 if $\text{size}({}^bD_{\text{total},k}) - \text{size}({}^bD_{\text{total},k-1}) \leq \delta_{\text{size}}$ **then**
 $n_{\text{extremities}} \leftarrow n_{\text{extremities}} + 1$
 ${}^bT_{k+1} \leftarrow \text{InitialViewpoint}()$
 else
 Fit a curve on the cumulative drawing's cloud.
 $\text{Curve} \leftarrow \text{Fit}({}^bD_{\text{total},k}, \text{FittingMethod})$
 Obtain next view pose from fit's extrapolation.
 ${}^b\mathbf{d}_{k+1} \leftarrow \text{Extrapolate}(\text{Curve}, \delta_{\text{distance}})$
 ${}^b\mathbf{R}_{k+1} \leftarrow \text{GetNormalVec}({}^b\mathbf{d}_{k+1}, {}^bC_{\text{total},k})$
 ${}^bT_{k+1} \leftarrow ({}^b\mathbf{d}_{k+1}, {}^b\mathbf{R}_{k+1})$
 $k \leftarrow k + 1$
return ${}^bD_{\text{total},k_{\text{stop}}}$

B. Constrained Next Best View (C-NBV) Algorithm

This planner restructures the NVP task into a search problem on a voxel grid. In brief, this grid is constructed from the point clouds and updated at each measurement. A frontier region is computed to constrain and generate the candidate search space, within which each candidate is scored with a viewpoint quality metric. The best candidate is selected as the next view. This repeats until a termination criterion is met.

The algorithm voxelizes the obtained processed clouds into an octree grid of occupancy probabilities. The grid distinguishes between occupied, unoccupied, and unknown cells based on their occupancy probabilities—respectively more than, less than, and equal to 0.5. We use OctoMap [46] as a probabilistic voxel occupancy grid. This grid allows efficient storage and querying of cell probabilities.

Let $\mathcal{G}_k$ be the voxel grid generated at step k . The local scene's cumulative cloud ${}^bC_{\text{total},k}$ is voxelized to map the currently available knowledge, while the reconstructed drawing's cumulative cloud ${}^bD_{\text{total},k}$ is used to identify those voxels belonging to the drawing. Since the drawing is the region of interest, we determine its frontier on the grid. Here we define a frontier cell to have at least one unknown neighbor and at least another neighbor belonging to the drawing's region. The frontier at step k is denoted by $\mathcal{F}_k$ and represents the boundary of current knowledge about the drawing used to determine high-vantage locations for generating viewpoint search spaces. This constrains the search at step k for the next view in search space $\mathcal{S}_k$ where candidate viewpoints are scored and ranked. The space is constructed by generating regions from geometric primitives (e.g., cubes, spheres) around each frontier cell, and then concatenating them such that $\mathcal{S}_k = \bigcup_{f \in \mathcal{F}_k} S(f)$ where $S(\cdot)$ generates a search space primitive for a single cell.

The camera's viewpoint s_k can be expressed in the base frame as ${}^b s_k = ({}^b x, {}^b y, {}^b z, \alpha_x, \alpha_y, \alpha_z)$, where $({}^b x, {}^b y, {}^b z)$ is the position in the grid, and $(\alpha_x, \alpha_y, \alpha_z)$ is the orientation obtained as anticlockwise rotations about the respective axes. It can also be expressed in the camera frame as ${}^c s_k = ({}^c x, {}^c y, {}^c z, \alpha_R, \alpha_P, \alpha_Y)$ where $({}^c x, {}^c y, {}^c z)$ is the position with respect to the camera's frame, and $(\alpha_R, \alpha_P, \alpha_Y)$ is the

orientation specified with respect to the local roll-pitch-yaw frame $(\hat{R}, \hat{P}, \hat{Y})$ about the camera's body (see Fig. 5). The transformations from ${}^c s_k$ to ${}^b s_k$ relate these expressions.

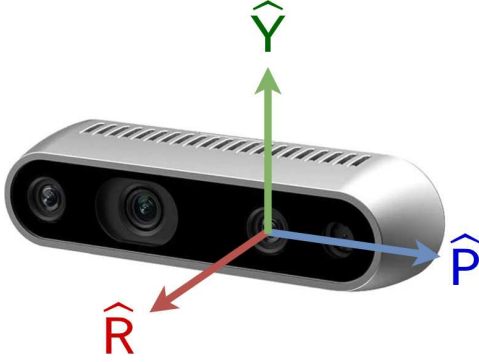


Fig. 5. Specification of the camera frame $(\hat{R}, \hat{P}, \hat{Y})$ for orientation. $\hat{R}$ is normal to the lens extending from the front. $\hat{P}$ extends horizontally from left to right on the camera's front body. $\hat{Y}$ is obtained binormally to the rest.

All candidates in S_k are scored using a viewpoint quality metric $Q(\cdot)$. The next view is set to the best-scoring candidate,

$${}^b T_{k+1} = \operatorname{argmax}_{s \in S_k} Q(s) \quad (3)$$

We note that ${}^b s_k$ represents the candidate viewpoints s in the base frame at step k of which the highest-scoring candidate $s^* = \operatorname{argmax}_{s \in S_k} Q(s)$ becomes the chosen next viewpoint ${}^b T_{k+1}$ to which the robot moves from steps k to $k+1$.

The quality of an candidate viewpoint at step k is given by,

$$Q(s_k) = \lambda \cdot \frac{\operatorname{gain}(s_k)}{\sum_{s \in S_k} \operatorname{gain}(s)} - (1 - \lambda) \cdot \frac{\operatorname{cost}(s_k)}{\sum_{s \in S_k} \operatorname{cost}(s)} \quad (4)$$

This is a convex combination of a gain term and a cost term, expressed as a proportion of their search space totals. The parameter $\lambda \in [0, 1]$ determines, for a particular viewpoint, the relative weight between its gain and cost terms $\operatorname{gain}(s_k) / \sum_{s \in S_k} \operatorname{gain}(s)$ and $\operatorname{cost}(s_k) / \sum_{s \in S_k} \operatorname{cost}(s)$, expressed as fractions of the candidate population totals—that is, the aggregate values for every $s \in S_k$. Here, $\operatorname{gain}(s_k)$ quantifies the relevant information gain obtained from choosing s_k as the next viewpoint. The function $\operatorname{cost}(s_k)$ offsets this gain and is the Euclidean distance between the current viewpoint and the candidate s_k . This penalizes the quality of a viewpoint on distance from the current position. Therefore, a smaller value for λ prioritizes the highest-ranking viewpoint candidates which are also near the current position.

Hence, the parameter λ directly affects the distance traveled within a measurement step and thus the exploration time. However, λ does not affect the NBV algorithms' coverage performance in terms of feature reconstruction. In practice, for highly constrained spaces, it is preferable that the robot explores in short bursts, thus favoring a reduced value for λ . Conversely, for highly spacious conditions, the distance constraints can be relaxed, allowing the robot to move to viewpoints further away, which in turn favors an increased value for λ . In our evaluations, the parameter is set to $\lambda = \frac{1}{2}$ to

give equal weight to the gain and cost terms, as the evaluation's focus is to assess the feature reconstruction capability. With that said, the parameter λ can be readjusted to application-specific and case-specific needs, if necessary.

The $\operatorname{gain}(\cdot)$ function is obtained by summing over the grid,

$$\operatorname{gain}(s_k) = \sum_{g \in \mathcal{G}_k} [h(g | s_k) \cdot p_\phi(g) \cdot p_v(g | s_k)] \quad (5)$$

Here, $h(g | s_k)$ is the information entropy decrease which measures absolute information gain at a cell g after placing the next viewpoint at s_k . The feature probability $p_\phi(g)$ estimates the feature membership of the cell g , i.e., the chances of belonging to the drawing. The visibility probability $p_v(g | s_k)$ estimates the chance that the cell g is visible from s_k . The probabilities $p_\phi(g)$ and $p_v(g | s_k)$ respectively penalize distance from the desired feature (the drawing) and occlusion.

The entropy decrease $h(g | s_k)$ is obtained through the entropy function $H(\cdot)$ used in information theory, as follows,

$$h(g | s_k) = H(o_g | s_{k-1}) - H(o_g | s_k) \quad (6)$$

The binary occupancy random variable o_g models whether the cell g is unoccupied ($o_g = 0$) or occupied ($o_g = 1$). The information entropy $H(o_g | s_k)$ quantifies the uncertainty of the binary random variable o_g denoting the occupancy of cell g . When this information entropy decreases across two consecutive measurements, i.e., when $h(g | s_k) > 0$, then we have “lost some uncertainty” or “gained information” about the occupancy state of the grid cell g , after the k^{th} measurement from cell s_k . This procedure is illustrated as shown in Fig 6.

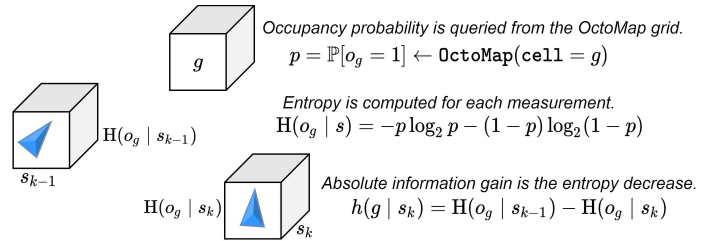


Fig. 6. The absolute information gained (about the occupancy state at a cell g) by moving from the viewpoint s_{k-1} to the viewpoint s_k corresponds to the decrease in information entropy about its occupancy state.

The feature probability is modeled with exponential decay,

$$p_\phi(g) = \mathbb{P}[\phi_g = 1] = \exp[-\alpha_\phi \operatorname{dist}_{\mathcal{F}_k}(g)^2] \quad (7)$$

The feature indicator random variable ϕ_g models whether the cell g belongs to the drawing ($\phi_g = 1$) or otherwise ($\phi_g = 0$). The function $\operatorname{dist}_{\mathcal{F}_k}(\cdot)$ returns the shortest Euclidean distance from g to the nearest frontier cell in $\mathcal{F}_k$. The parameter $\alpha_\phi > 0$ is used to tune the exponential decay profile. Maximizing only the absolute information gain between two consecutive viewpoints would lead to seeking novel information about the state of the grid regardless of its relevance to the desired feature. Therefore, new information gained about each cell must also be scaled by the cell's probability of belonging to the desired feature. This feature probability p_ϕ is approximated using the assumption that cells near the desired feature have a higher probability of belonging to the feature. This assumption is modeled using an exponential decay profile applied onto

the distance to the nearest frontier cell. We illustrate this computation as shown in Fig. 7.

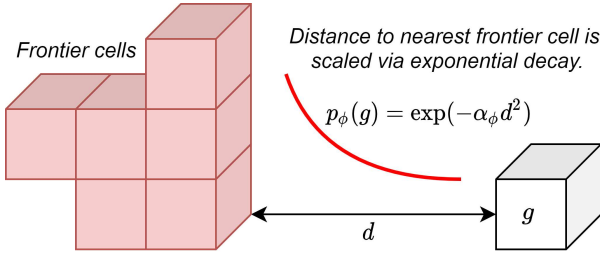


Fig. 7. The feature probability approximates the probability that a grid cell g belongs to the desired feature. The frontier cells in this case represent the cells that have been determined to be part of the desired feature.

The visibility probability at cell g from candidate s_k is,

$$p_v(g | s_k) = \mathbb{P}[v_{g,s_k} = 1] = \prod_{r \in \mathcal{R}_k} \mathbb{P}[o_r = 0] \quad (8)$$

The binary visibility random variable v_{g,s_k} models whether the cell g is visible ($v_{g,s_k} = 1$) or otherwise ($v_{g,s_k} = 0$) from the candidate s_k . An unobstructed view to the cell g from a candidate s_k must be preferred. Raycasting is performed from s_k to g , where $\mathcal{R}_k$ contains all cells traversed by the ray. The probabilities that the ray cells are unoccluded $\mathbb{P}[o_r = 0]$ are multiplied to yield $p_v(g | s_k)$, i.e., the probability that the cell g is visible from the candidate viewpoint s_k . The above procedure is illustrated as shown in Fig. 8.

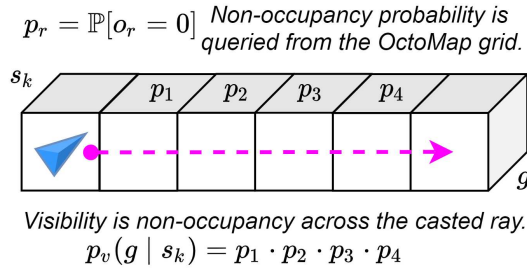


Fig. 8. The visibility probability of a cell g from the candidate viewpoint s_k is computed as the probability p_v that all their intermediate cells are unoccupied.

During the search, all grid positions ($^b x, ^b y, ^b z$) within S_k are attempted for scoring. For each candidate position, the orientation is searched by varying the angles α_P and α_Y . The C-NBV planner relaxes the angle α_R to not overly constrain the motion planner, since the normal direction about the lens has a comparably lesser effect on the information gain in the acquired image. After determining the next viewpoint s_k^* , the camera's grid position (x, y, z) and orientation α_P and α_Y are set accordingly. The angle α_R is determined by the motion planner, and the camera is moved to its new pose.

With the quality metric $Q(\cdot)$ fully defined, a next viewpoint can be obtained, and the procedure elaborated above repeats until the following stop condition is met:

$$[\max_{s \in S_k} Q(s) < \delta_Q] \vee [\text{card}(\mathcal{F}_k) = 0] \quad (9)$$

That is, the routine stops once the optimal viewpoint quality falls below a certain threshold δ_Q , or when the number of

frontier cells reaches zero, whichever condition is met first. This procedure is outlined in Algorithm 2.

This formulation is based on the object reconstruction algorithm developed in [12] as well as the information gain approach in [13]. However, these formulations are not well-adapted to our problem, and are either impractical (search spaces are too large and NBV selection is too slow) or infeasible (scrap objects are often too large to fully scan).

As presented above, we implement several modifications to reformulate these approaches for our requirements. We modify the frontier definition to focus on the desired feature. We also generate constrained search spaces around specific frontier cells to vastly decrease the search's time and memory costs. To determine the NBV's orientation, we constrain its range by pointing the camera at the frontier cells.

Algorithm 2: Constrained Next Best View (C-NBV)

Input : Initial view containing part of the drawing.
Output: $^b D_{\text{total}, k_{\text{stop}}}$, fully-reconstructed drawing cloud.
Initialize step, stopping condition, and viewpoint.
 step $k \leftarrow 0$
 stop $\leftarrow \text{False}$
 $^b T_0 \leftarrow \text{InitialViewpoint}()$
Search the grid until viewpoint quality is exhausted.
while stop $\equiv \text{False}$ **do**
 Acquire, transform, concatenate, and filter clouds.
 $(^b D_{\text{total}, k}, ^b C_{\text{total}, k}) \leftarrow \text{ProcessClouds}(k)$
 Generate octree grid from cumulative clouds.
 $\mathcal{G}_k \leftarrow \text{GenerateGrid}(^b C_{\text{total}, k}, ^b D_{\text{total}, k})$
 Determine and filter the frontier cells in the grid.
 $\mathcal{F}_k \leftarrow \text{GetFrontier}(\mathcal{G}_k)$
 Generate viewpoint search space around frontier.
 $S_k \leftarrow \bigcup_{f \in \mathcal{F}_k} S(f)$
 Obtain next viewpoint with highest quality.
 $^b T_{k+1} \leftarrow \arg\max_{s \in S_k} Q(s)$
 Evaluate and update the stopping condition.
 stop $\leftarrow \max_{s \in S_k} Q(s) < \delta_Q$ **or** $\text{card}(\mathcal{F}_k) = 0$
 $k \leftarrow k + 1$
return $^b D_{\text{total}, k_{\text{stop}}}$

C. Guided Next Best View (G-NBV) Algorithm

This planner implements specialized modifications to the previous algorithm to radically increase performance. It does this by not only constraining the search, but also explicitly guiding it along the feature. The planner thus uses feature information to perform greedy optimizations.

Unlike the E-NVP planner, this algorithm does not assume that it explores a path. However, by guiding its search along the feature, it retains the performance advantages of path exploration. Yet, its probabilistic NBV formulation is more robust against unknown space, occlusions, and adversarial geometries. In a sense, this carries the performance advantages of the path exploration paradigm and the robustness characteristics of the probabilistic NBV formulation.

We repeat the formulation of the C-NBV algorithm until after the frontier $\mathcal{F}_k$ is determined. Despite the large per-

formance improvements of the already reduced frontier, the search space can still become quite large. Consider the scenario of a large object with a drawing traversing long parts of its surface. This produces a long frontier with little overlap between the individual search spaces $S(\cdot)$, causing the concatenated search space S_k to significantly grow.

To address this, the G-NBV planner reduces the frontier $\mathcal{F}_k$ into a single frontier cell $\hat{f}_k$. For this, the frontier cells are clustered using a suitable voxel clustering method. We use a connectedness-based clustering method, whereby any neighboring frontier cells are lumped into the same cluster. Now, the frontier is composed of several clusters, of which the nearest one (to the current pose) is chosen. The centroid $\hat{f}_k$ of this nearest cluster is computed and then used as the single-voxel frontier for generating the search space.

This offers numerous performance advantages. First, by trimming the other clusters, the planner explores any number of branches, one at a time, avoiding long and exhaustive searches. Second, since there is only one frontier cell, then the entire search space consists of a single geometric primitive around the cell, *e.g.*, a sphere around the centroid. This is a reasonable step, as frontier cells tend to surround the current reconstructed drawing's endpoints. Thus, searching around this centroid would implicitly guide the planner along the drawing.

A third advantage lies in determining orientations for the candidate viewpoints. This is normally expensive as it drastically increases the search space, since for every position there are several the camera orientations. We eliminate this problem entirely by constraining the orientation from the candidate cell s to the frontier centroid $\hat{f}_k$, thus predetermining the values for the angles α_P and α_Y . This reduces G-NBV's search to only the grid positions, since α_P and α_Y are predetermined and α_R is delegated to the motion planner. The search space is thus not only reduced cell-wise, *i.e.*, attempt only the search space around $\hat{f}_k$ rather than around the entire frontier, but also pose-wise, *i.e.*, search only grid positions in the search space while constraining the camera orientation.

This constraint is justified due to the localized information on the drawing. In effect, high viewpoint qualities concentrate in the unknown space around the current reconstructed drawing's endpoints. Fixing the orientation towards the centroid eliminates the exploration of alternatives where the gain is often marginal or negative. By exploiting the drawing's structure, the planner is able to rapidly select the next viewpoint.

The G-NBV routine is summarized in Algorithm 3.

D. Cutting Path Generation

By using any of the three aforementioned algorithms, we obtain the fully-reconstructed cloud of the drawing. This cloud is used to generate a suitable cutting path (along the drawing) that can eventually be used as a reference for cutting control. We accomplish this task of converting unstructured point clouds to ordered paths, as in the mapping (2), by using suitable curve fitting methods. This is the same class of methods used in the Extrapolated NVP's curve fitting step. In effect, either of the corresponding methods (NURBS-based point-distance minimization, or topological skeletonization) as

Algorithm 3: Guided Next Best View (G-NBV)

Input : Initial view containing part of the drawing.

Output: ${}^bD_{\text{total},k_{\text{stop}}}$, fully-reconstructed drawing cloud.

Initialize step, stopping condition, and viewpoint.

step $k \leftarrow 0$

stop $\leftarrow \text{False}$

${}^bT_0 \leftarrow \text{InitialViewpoint}()$

Search the grid until viewpoint quality is exhausted.

while stop $\equiv \text{False}$ **do**

Acquire, transform, concatenate, and filter clouds.

$({}^bD_{\text{total},k}, {}^bC_{\text{total},k}) \leftarrow \text{ProcessClouds}(k)$

Generate octree grid from cumulative clouds.

$\mathcal{G}_k \leftarrow \text{GenerateGrid}({}^bC_{\text{total},k}, {}^bD_{\text{total},k})$

Determine and filter the frontier cells in the grid.

$\mathcal{F}_k \leftarrow \text{GetFrontier}(\mathcal{G}_k)$

Reduce the frontier set into a single frontier cell.

$\mathcal{F}_k \leftarrow \text{GetClusters}(\mathcal{F}_k, \text{ClusteringMethod})$

$\mathcal{F}_k \leftarrow \text{GetNearestCluster}(\mathcal{F}_k)$

$\hat{f}_k \leftarrow \text{GetClusterCentroid}(\mathcal{F}_k)$

Generate viewpoint search space around centroid.

$S_k \leftarrow S(\hat{f}_k)$

Obtain next viewpoint with orientation constraint.

${}^bT_{k+1} \leftarrow \underset{s \in S_k}{\text{argmax}} Q(s) \text{ s.t. } {}^bR_{k+1} = \text{Rot}(s, \hat{f}_k)$

Evaluate and update the stopping condition.

 stop $\leftarrow \max_{s \in S_k} Q(s) < \delta_Q$ **or** $\text{card}(\mathcal{F}_k) = 0$

$k \leftarrow k + 1$

return ${}^bD_{\text{total},k_{\text{stop}}}$

explained in subsection V-A is suitable for obtaining a cutting path from the fully-reconstructed drawing.

VI. SIMULATIONS & PHYSICAL EXPERIMENTS

This section details the approaches taken to evaluate and compare our NVP algorithms in simulation and in the physical experiments. Additionally, it defines the benchmarking metrics and ground truths used for performance quantification.

A. Design of Realistic Simulation

The simulation is configured to mimic realistic conditions and outcomes. In effect, the simulation's camera model replicates the sensing properties of the D435 stereocamera used in the experiment, *e.g.*, image size, and sensory noise modeled after the D435's noise distribution found in its datasheet [47].

For a realistic evaluation of the NVP algorithms, the test objects must carry adversarial features commonly found in the scrapyard. We select six feature categories or challenges with which we evaluate our planners. These are:

- 1) Smooth surface without occlusion
- 2) Sharp occlusion between smooth surfaces
- 3) Smooth occlusion along a smooth surface
- 4) Sharp transition between smooth surfaces
- 5) Highly nonsmooth surface with occlusions
- 6) Highly discontinuous surface

For this, we acquire high-detail 3-D scans of actual metal scrap pieces (see Table I) adequately selected and retrieved

TABLE II
SIMULATION RESULTS: EVALUATING BENCHMARKING METRICS FOR EACH NVP ALGORITHM ON ALL OBJECT CATEGORIES

Challenge (category)	Planner (algorithm)	Viewpoints (steps)	Total Duration (sec)	Avg. Step Duration (sec/step)	Total Displacement (cm)	Coverage (voxels)	Coverage (%)
Smooth Cylinder <i>Category 1</i>	Extrapolated NVP	9	61.9	6.9	144.6	197/199	99.0
	Constrained NBV	4	808.0	202.0	48.5	197/199	99.0
	Guided NBV	4	38.6	9.6	67.8	197/199	99.0
Sharp Cylinder <i>Category 2</i>	Extrapolated NVP	8	43.4	5.4	88.3	372/569	65.4
	Constrained NBV	4	417.5	104.4	87.7	569/569	100
	Guided NBV	3	43.4	14.5	66.0	568/569	99.8
Round Tank <i>Category 3</i>	Extrapolated NVP	6	30.1	5.0	149.5	82/82	100
	Constrained NBV	3	153.3	51.1	81.8	82/82	100
	Guided NBV	3	41.6	13.9	68.1	82/82	100
T-Piece <i>Category 4</i>	Extrapolated NVP	4	18.0	4.5	81.7	179/182	98.4
	Constrained NBV	4	463.5	115.9	88.7	179/182	98.4
	Guided NBV	3	47.0	15.7	53.7	177/182	97.25
I-Beam <i>Category 5</i>	Extrapolated NVP	6	29.0	4.8	90.0	234/532	44.0
	Constrained NBV	5	549.0	109.8	108.4	525/532	98.7
	Guided NBV	3	56.4	18.8	10.5	501/532	94.2
Basket <i>Category 6</i>	Extrapolated NVP	8	46.0	5.7	94.1	153/155	98.7
	Constrained NBV	2	187.7	93.8	11.82	130/155	83.9
	Guided NBV	2	50.6	25.3	23.9	145/155	93.5

Note: The Extrapolated NVP planner backtracks to the initial point after finding the first endpoint. As a result, it records an additional step: the revisited viewpoint.

TABLE III
VISUALIZING VIEWPOINTS, DISPLACEMENT, AND COVERAGE FOR EACH NVP AGAINST FOUR SIMULATED OBJECTS

	Algorithm 1: E-NVP		Algorithm 2: C-NBV		Algorithm 3: G-NBV	
	Viewpoints	Coverage	Viewpoints	Coverage	Viewpoints	Coverage
Sharp Cylinder						
Round Tank						
I-Beam						
Basket						

The magenta graph shows the displacement between views while the green rays show their poses. For coverage, recovered voxels are in green while missed ones are in red.

from our industrial collaborator's shipbreaking yard. We add two reference objects (cylinders) to examine the robustness of each algorithm against a simple sharp self-occlusion.

B. Setup of Physical Experiment

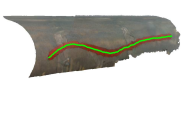
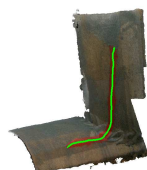
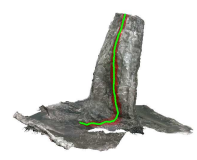
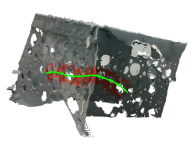
The evaluation steps are standardized and replicated in both the simulated and experimental environments. For this, both environments are configured to reduce discrepancies between their conditions. The same red color for the drawing is used on both the simulation objects and the physical objects.

TABLE IV
EXPERIMENTAL RESULTS: EVALUATING BENCHMARKING METRICS FOR EACH NVP ALGORITHM ON ALL OBJECT CATEGORIES

Challenge (category)	Planner (algorithm)	Viewpoints (steps)	Total Duration (sec)	Avg. Step Duration (sec/step)	Total Displacement (cm)	Coverage (voxels)	Coverage (%)
Smooth Cylinder Category 1	Extrapolated NVP	8	24.3	3.0	119.5	265/270	98.1
	Constrained NBV	1	31.4	31.4	21.7	268/270	99.3
	Guided NBV	1	18.7	18.7	22.3	267/270	98.9
Sharp Cylinder Category 2	Extrapolated NVP	5	19.4	3.9	51.9	144/167	86.2
	Constrained NBV	3	561.2	187.1	93.7	162/167	97.0
	Guided NBV	2	68.1	34.1	62.0	165/167	98.8
Round Tank Category 3	Extrapolated NVP	4	13.1	3.3	67.3	324/329	98.5
	Constrained NBV	3	252.4	84.1	89.4	320/329	97.3
	Guided NBV	3	106.2	35.4	63.8	314/329	95.4
T-Piece Category 4	Extrapolated NVP	5	25.0	5.0	29.5	109/111	98.2
	Constrained NBV	2	178.5	89.3	44.7	110/111	99.1
	Guided NBV	1	28.4	28.4	38.7	111/111	100
I-Beam Category 5	Extrapolated NVP	1	9.6	9.6	28.1	92/425	21.6
	Constrained NBV	6	1015.8	169.3	122.9	410/425	96.5
	Guided NBV	7	153.3	21.9	167.7	407/425	95.8
Basket Category 6	Extrapolated NVP	4	6.8	1.7	32.3	117/171	68.4
	Constrained NBV	4	358.4	89.6	105.3	163/171	95.3
	Guided NBV	3	112.0	37.4	91.2	166/171	97.1

Note: The Extrapolated NVP planner backtracks to the initial point after finding the first endpoint. As a result, it records an additional step: the revisited viewpoint.

TABLE V
EXPERIMENTAL OUTPUT: RECONSTRUCTIONS USING THE GUIDED NBV PLANNER AND CORRESPONDING GENERATED CUTTING PATHS

Features	Category 1 Smooth Cylinder Smooth surface	Category 2 Sharp Cylinder Sharp occlusion	Category 3 Round Tank Smooth occlusion	Category 4 T-Piece Sharp transition	Category 5 I-Beam Nonsmooth surface	Category 6 Basket Discontinuous surface
Local Reconstruction						
Drawing Reconstruction						
Generated Cutting Path						

Furthermore, the camera settings and the ambient lighting are kept consistent and maintained throughout the experiments. In addition, the planners are evaluated and tuned fairly. All planners share the same filter ξ , E-NVP uses $\delta_{\text{distance}} = 1$ voxel, and both NBV planners share the parameters ($\lambda = \frac{1}{2}$, $\alpha_\phi = 5$).

More importantly, the physical test objects are direct feature analogs of the simulated ones as shown in Table I. This means that the NVP algorithms evaluated in either the simulations or the experiments face the same types of challenges in each feature category. For these reasons, the simulation and experimental results may be used collectively to reliably draw conclusions on the performance of the NVP algorithms.

Note that for both evaluations, the objects were fixed and

evaluated one at a time. That is, for each object we first obtain a ground truth of the drawing using the method at the end of this section. Afterwards, the three NVP algorithms are run sequentially and all their results are collected in the same frame without moving the tested object. We repeat these four procedures with every object until completion.

C. Benchmarking Metrics

The following metrics are chosen to assess and quantify the efficiency and robustness of our viewpoint planners.

1) *Number of Viewpoints*: This counts the number of iterations in the exploration routine, which corresponds to the number of viewpoints visited until termination. This excludes

the initial viewpoint, since it is an input to the exploration task used to generate the first round of viewpoint candidates. For instance, $n_{\text{viewpoints}} = 1$ means the robot moved to one viewpoint from the initial pose, reached the stopping condition, and then terminated the search.

This metric helps compare the number of steps taken by each NVP algorithm until termination.

2) *Total Duration*: In our evaluations, we measure only the viewpoint planning time and exclude motion planning and execution times as these are external processes. In other words, we partition the total exploration time into NVP processing time and motion-related time. Let Δt be this next view planning time which we call “Total Duration” in Tables II and IV. In our evaluations, this is computed as follows,

$$\Delta t = \Delta t_{\text{exploration}} - \Delta t_{\text{motion}} \quad (10)$$

This metric allows us to directly compare the processing duration spent by each NVP algorithm from the start of the exploration task until termination.

3) *Average Step Duration*: It is desirable to assess and compare the average processing time per iteration $\overline{\Delta t}$ taken by each NVP algorithm for a particular scenario. This is computed using the previous two metrics, as follows,

$$\overline{\Delta t} = \Delta t / n_{\text{viewpoints}} \quad (11)$$

This allows us to more easily perceive the trade-off between number of steps and step duration across each algorithm.

4) *Total Displacement*: We measure the spatial configuration of the chosen viewpoints independent of the robot’s motion plan, by summing the displacement magnitudes between viewpoints. Specifically, let $(p_k)_{k=0}^{n_{\text{viewpoints}}}$ be the sequence of viewpoint positions, indexed in the order they were visited. This includes the initial viewpoint, since we sum the magnitudes between viewpoints. The total displacement d_{total} is,

$$d_{\text{total}} = \sum_{k=1}^{n_{\text{viewpoints}}} \|p_k - p_{k-1}\|_2, \quad (12)$$

This allows us to compare the transient behavior of each algorithm, alongside the magenta polygonal chain connecting all visited viewpoints in Table V.

5) *Coverage*: We use point cloud coverage as a measure of correctness for the reconstructed drawings resulting from the use of each NVP algorithm. For our evaluations, we define the coverage of a cloud bC over a ground truth cloud bG as,

$$\text{coverage}_G({}^bC) = \frac{\text{card} [\text{vox}({}^bC) \cap \text{vox}({}^bG)]}{\text{card} [\text{vox}({}^bG)]} \quad (13)$$

Here $\text{vox}(\cdot)$ voxelizes the clouds in a common and precise grid to determine their overlap, and $\text{card}(\cdot)$ is the cardinality which returns the number of voxels. The cloud bG represents the ground truth to which bC is compared and over which coverage is obtained. We emphasize that both clouds must be expressed with respect to a common base frame b .

In our evaluations, bC is the final reconstructed drawing ${}^bD_{\text{total}, k_{\text{stop}}}$ and bG is accepted to represent the actual drawing in point cloud form. We discuss the method used to generate

bG in the next subsection. We report coverage results in Tables II and IV in two formats. The first shows the number of voxels in the ground truth cloud, while the second is a percentage.

D. Ground Truth Generation

For the purposes of coverage calculation, the ground truth cloud bG ideally represents a perfect reconstruction of the drawing in the form of a point cloud. We must obtain a good approximation for this ideal within the limitations of our stereocamera. For this, we must scan the object’s local region which contains the drawing. The NVP algorithms are meant to automate this procedure of picking viewpoints. However, to improve the ground truth, we manually pick good viewpoints for the robot. The robot stays in place at each viewpoint to scan the same region repeatedly. The obtained images are averaged until the discrepancy between iterates of this average cloud falls below a preset threshold. This threshold is expressed as a proportion of the occupancy grid’s voxel size. This procedure is briefly sketched in the following pseudocode.

```

define GenerateGroundTruth():
    Initialize ground truth cloud  ${}^bG$ 
    while UserInput():
        Move the robot to the next viewpoint
        Initialize cumulative moving average (CMA)
        while Error < Threshold:
            Acquire image point cloud
            Update CMA for this viewpoint
            Update error between last two CMAs
            Concatenate latest CMA with  ${}^bG$ 
    return  ${}^bG$ 

```

We use this user-assisted procedure in both simulations and experiments to generate near-optimal drawing reconstructions for each test case within the sensory limitations of the stereocamera. This reconstruction has very little noise and can serve as a ground truth for coverage calculations.

VII. RESULTS

In this section, we present and discuss the results obtained from the evaluations in simulations and experiments.

We tabulate our results for the aforementioned benchmarking metrics in Table II for the simulations and Table IV for the experiments. In addition, in Table III we visualize for each planner, some of its simulation results: viewpoint displacement, poses, and output cloud coverages. Finally, in Table V we display the experimental output of our exploration and reconstruction task (using G-NBV for planning) against each physical test object. We show the full reconstruction of the desired feature to demonstrate the effectiveness of our feature-based NVP paradigms in a real-world scenario.

We discuss the significance of our results and the nuances between our planners in the following subsections.

A. Exploration Efficiency

The planners exhibit different transient properties while exploring each object category. In general, we observe that the

E-NVP average step duration is shorter—fitting and extrapolation is relatively quick—yet the planner often requires more iterations to fully explore the feature. In contrast, the NBV planners finish in less views but often take more processing time per step to search their grids.

For E-NVP, the average step durations are fairly consistent during successful runs across object categories. This means that the E-NVP total duration is largely affected by the number of steps in the exploration routine. This implies that its total duration is grows with the feature's size on the object: longer drawings on larger objects require more steps to be fully explored and thus increase total duration. This is observed for object category 1 which features the longest drawing. This sensitivity to feature size is expected, since E-NVP typically gains less information per step when compared to the NBV planners. Both the C-NBV and the G-NBV planners are less affected by feature size (meaning number of steps) and more by the total grid size (meaning average step duration) and by the scene's complexity (*e.g.*, in categories 2, 5, and 6).

The total displacements and viewpoint poses reveal that E-NVP plans in a more conservative and predictable manner and can be seen to move along the drawing. By contrast, the NBV planners tend to pick views with much larger information gain without much consideration for their positions.

In terms of total duration, the G-NBV planner is shown to be up to around ten times faster than the C-NBV, especially in large scenes requiring a larger grid. This is expected since C-NBV's frontier definition and search space generation make it much more sensitive to grid size. Even with its feature-based constraints, the C-NBV planner is the slowest in every category. This further justifies the greedy and aggressive constraints used by G-NBV to speed up exploration.

B. Robustness against Adversarial Features

Adversarial features such as occlusions, nonsmoothness, and discontinuities affect the reconstruction effectiveness of each planner differently. In the absence of adversarial features (category 1) all methods perform quite well. Even with smooth occlusions (category 3) and unoccluded sharp transitions (category 4), coverage scores are almost perfect for all planners, and the drawing is fully reconstructed.

However, we observe coverage degradation in certain scenarios. E-NVP fails to fully reconstruct the drawing against sharp occlusions (category 2). The coverage score of 86.2% on category 2 in the experimental results is not to be interpreted as high, since it missed the portion of the drawing behind the occlusion anyway, meaning this is still considered a reconstruction failure. Both NBV planners are unaffected by difficult occlusions and manage to fully reconstruct the drawing in categories 2 and 5. We also see this on the sharply occluded portion of the experiment's basket (category six), where E-NVP failed only on the occluded portion, while the NBV planners manage to overcome it and fully scan the drawing.

On that note, all planners perform quite well against surface discontinuities where voxel sizes are increased to reduce the disturbance inflicted by the gaps. The lower score of 83.9% for C-NBV in the simulation's category 6 is not due to surface discontinuities, but to grazing incidence.

This effect occurs when the ray incidence is nearly parallel to the object surface. Grazing incidence dilutes the sampling density at which point the image obtained is noisier, distorted, and poorer in information. This effect occurs in the simulation's category 6 for both NBV planners, as can be seen by the viewpoint poses and the suffering coverage near the extremities. The NBV planners may suffer suboptimal viewing angles since this optical disturbance is unmodeled by their viewpoint quality metric. E-NVP avoids this problem by maintaining an orientation quasi-normal to the surface and thus guarantees better viewing angles. Finally, we note that both NBV planners achieve similar coverage, meaning that the greedy approach of G-NBV comes at little coverage costs.

C. Planner Preference and Selection

To summarize the behavior of each planner, E-NVP extrapolates from the feature, C-NBV's search is constrained by the feature, and G-NBV's search is guided by the feature. Furthermore, each planner exhibits its own transient characteristics, strengths, and failure modes, which make it perform better or worse in particular scenarios.

If fast exploration is not a requirement, it is appropriate to use the C-NBV planner when the feature and scene are more complicated. The C-NBV planner searches more exhaustively due to its larger frontier and search spaces. Using G-NBV in exceedingly complicated scenes may yield the typical disadvantages of greedy optimization.

For simpler and smaller objects, E-NVP iterates quickly, handles gaps exceptionally well, and does not exhibit grazing incidence. Also, E-NVP works more predictably in confined spaces as its viewpoint displacement follows the drawing more conservatively, which is not guaranteed by the NBV planners.

E-NVP should be avoided when the application faces frequent occlusions and exotic surfaces. Occlusions trigger premature termination for E-NVP, as it poorly distinguishes between surface edges and drawing extremities. Meanwhile, objects with contorted surfaces may cause reachability issues with the E-NVP planner which tries to remain normal to the surface. The NBV planners overcome both issues by exploring more permissively. E-NVP is also limited to two extremities per feature. The NBV planners handle branches arbitrarily. G-NBV's frontier reduction makes it especially suited for this.

While each of the three algorithms exhibits particular strengths and weaknesses, G-NBV offers well-rounded advantages in terms of efficient scanning and effective drawing reconstruction, making it most suitable for our domain application in metal recycling. For other applications requiring feature-driven exploration, it is best to study the particularities of each problem to select a more appropriate NVP algorithm.

VIII. DISCUSSION

In this section, we discuss specific application scenarios of the proposed methods and the associated parameters.

A. Performance against Branched Drawings

In a shipbreaking environment, the cuts are kept as simple as possible to maintain safety and control over the cutting

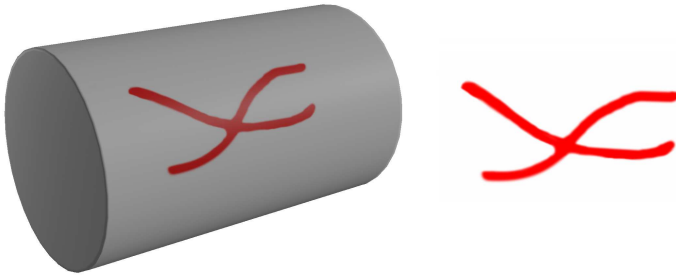


Fig. 9. Test object featuring a branched drawing with four extremities.

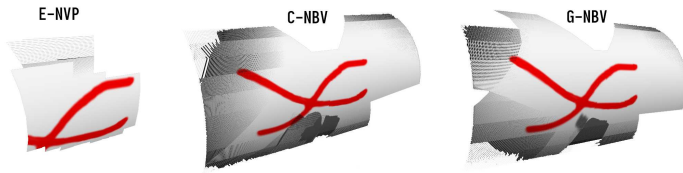


Fig. 10. Local surface reconstructions around the branched drawing.

operation and its outcomes. Although simpler cutting paths are preferred, it is still worthwhile to test against branched drawings—that is, drawings with more than two extremities.

The E-NVP method explicitly models the drawing as a curve and by design expects—and is therefore limited to—two extremities and thus cannot operate on branching curves. In contrast, the C-NBV and G-NBV methods operate at the voxel level. Accordingly, they have no conception of a branch (nor of a curve), but instead search through individual voxels in the space. This suggests that the NBV methods are capable of handling any number of branches, since as mentioned, they process the surface as voxels, regardless of branching. These algorithmic distinctions can be verified in the pseudocode listings of each of the methods in Section V.

We demonstrate the above remarks in simulation on a test object with a 4-extremity branched drawing. The test object (a simple cylinder) and its branched drawing are shown in Fig. 9. We test each of the viewpoint planners against this test object.

The local surface reconstruction of each method is shown in Fig. 10. As expected, the E-NVP recovers only two extremities of the branched drawing. In contrast, both the C-NBV and the G-NBV methods, recover the entirety of the branched drawing.

B. Behavior of the NBV Algorithms near Corners

Frontier-based NBV approaches may sometimes risk trapping the viewpoints near a surface corner. The purpose of our NBV algorithms is to map the object’s local surface containing the desired feature, not the full object nor its surroundings. In practice, the robotic arm almost never needs to explore the object surface close enough that it risks trapping the camera in a corner. Nevertheless, we discuss how such a scenario may occur and how to avoid it. While our NBV algorithms maximize (per viewpoint) the information gain relevant to the desired feature, there is a distance tradeoff between absolute information gained and relevant feature information:

- 1) To maximize the information gained per viewpoint, the camera is incentivized to be placed away from the object surface. This avoids corner fixation.

- 2) To capture relevant feature information, the camera must remain close enough to the desired feature. This may lead to corner fixation.

Our NBV algorithms’ tendency to avoid corner fixation is related to its preferred distance from the surface, which can be tuned using the decay factor α_ϕ in (7). By increasing α_ϕ , viewpoints closer to the surface are preferred. Conversely, lowering α_ϕ prefers more distant viewpoints. The tendency to avoid being trapped in corners can therefore be amplified or relaxed by decreasing or increasing α_ϕ , respectively. A scenario where the camera may get trapped in a corner would involve the desired feature passing through deep corners of very large objects. By sufficiently lowering α_ϕ , the planner can avoid trapping the camera in a corner.

IX. CONCLUSION

This work develops a feature exploration and reconstruction methodology for exploring an unknown feature carrying a desired characteristic (*e.g.*, a color), located on an unknown object’s surface, which lies in an unknown scene. This component serves a broader robotic system designed for automated cutting in metal scrap recycling. The component requires only an initial view containing a portion of the desired feature, and is expected to fully explore and reconstruct it. The underlying methodology is developed around next view planning to determine the next viewpoint given current scene information.

For this, we present three feature-driven next view planners that exploit the feature information collected from the scene, to rapidly and adeptly plan the next view. The first planner relies on fitting a curve to the feature from which the next view is extrapolated. The second is a probabilistic formulation on a voxel occupancy grid on which the search for the next view is constrained using feature information. The third repurposes the previous algorithm with greedy optimizations to guide the search directly via feature information. We evaluate our planners both in simulation and experiments and discuss their notable strengths and weaknesses. We note that our third planner is most suited for scrap cutting. Nevertheless, the other planners retain advantages in specific scenarios. We believe that the feature-driven paradigm for exploration as examined in this work can also be useful outside of scrap cutting for applications which require feature reconstruction in uncertain environments. In future work, we plan to continue developing components for our robotic cutting system, such as, vision-based torch-cutting control.

ACKNOWLEDGMENT

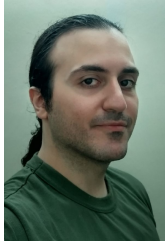
The authors would like to thank Roger Morton from EMR Group for his guidance in the metal scrap recycling industry.

REFERENCES

- [1] O. Dinu and A. M. Ilie, “Maritime vessel obsolescence, life cycle cost and design service life,” *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 95, p. 012067, nov 2015. [Online]. Available: <https://doi.org/10.1088/1757-899x/95/1/012067>
- [2] M. J. Kaiser, “A Review of Ship Breaking and Rig Scrapping in the Gulf of Mexico,” *Ocean Dev. & Int. Law*, vol. 39, no. 2, pp. 178–199, 2008. [Online]. Available: <https://doi.org/10.1080/00908320802013701>

- [3] W. Shen and G. Xing, "Study on status quo of shipbreaking sector and strategies," *2017 3rd Int. Conf. Energy, Environ. Mater. Sci. (EEMS 2017)*, vol. 94, p. 12163, nov 2017. [Online]. Available: <https://doi.org/10.1088/1755-1315/94/1/012163>
- [4] S. M. M. Rahman, J. Kim, and B. Laratte, "Disruption in Circularity? Impact analysis of COVID-19 on ship recycling using Weibull tonnage estimation and scenario analysis method," *Resour. Conserv. Recycl.*, vol. 164, pp. 105–139, 2021. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0921344920304560>
- [5] R. Bogue, "Cutting robots: A review of technologies and applications," *Ind. Rob.*, vol. 35, no. 5, pp. 390–396, 2008.
- [6] S. Gasper and J. Bickendorf, "Robot cutting in ship building industry - A new flexible approach for linking parametric design and fully automatic robot programming," in *ISR 2010 41st Int. Symp. Robot. Robot. 2010 (6th Ger. Conf. Robot.)*, 2010, pp. 1–4.
- [7] J. Bickendorf, "New Applications of Cutting and Welding Robots with Automatic Offline-Programming," in *Proc. ISR 2016 47th Int. Symp. Robot.*, 2016, pp. 1–6.
- [8] B. Denkena and T. Lepper, "Enabling an Industrial Robot for Metal Cutting Operations," *Procedia CIRP*, vol. 35, pp. 79–84, 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.procir.2015.08.100>
- [9] R. M. Arner, J. Beard, A. N. Cuneo, J. R. Dydo, J. H. Gaffney, T. Holverson, J. E. Jones, M. D. Mann, and V. L. Rhoades, "Hybrid Induction Plasma-Oxygen Cutting, Using a Mobile Robot System for Shipyard Application," in *2016 SNAME Marit. Conv.*, 2016, p. 103. [Online]. Available: <https://onepetro.org/SNAMESMC/proceedings-abstract/SMC16/3-SMC16/D033S010R006/21246>
- [10] K. S. Yoo, H. R. Ryu, and C. Choi, "Control architecture design for an gas cutting robot," *Proc. 2008 2nd Int. Conf. Futur. Gener. Commun. Networking, FGCN 2008*, vol. 4, pp. 66–71, 2008.
- [11] J. Akl, F. Alladkani, and B. Calli, "Towards Robotic Metal Scrap Cutting: A Novel Workflow and Pipeline for Cutting Path Generation," in *2021 IEEE 17th Int. Conf. Autom. Sci. Eng.*, 2021, pp. 132–137.
- [12] J. Daudelin and M. Campbell, "An Adaptable, Probabilistic, Next-Best View Algorithm for Reconstruction of Unknown 3-D Objects," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1540–1547, 2017.
- [13] S. Isler, R. Sabzevari, J. Delmerico, and D. Scaramuzza, "An information gain formulation for active volumetric 3D reconstruction," in *2016 IEEE Int. Conf. Robot. Autom.*, 2016, pp. 3477–3484.
- [14] H. Mei, Z. Li, and C. Zou, "Automatic Cutting System Design of Robot Hand Based on Stereo Vision," *Sci. Program.*, vol. 2022, p. 4663213, 2022. [Online]. Available: <https://doi.org/10.1155/2022/4663213>
- [15] Z. Xu, M. Liang, X. Fang, G. Wu, N. Chen, and Y. Song, "Research on Autonomous Cutting Method of Cantilever Roadheader," *Energies*, vol. 15, no. 17, 2022. [Online]. Available: <https://www.mdpi.com/1996-1073/15/17/6190>
- [16] D. Lindberget, "Automatic generation of robot targets : A first step towards a flexible robotic solution for cutting customized mesh tray," p. 34, 2022.
- [17] J. Cai, Z. Ding, Y. Zhang, and M. Liu, "Trajectory planning and simulation for intersecting line cutting of the industry robot," *Proc. World Congr. Intell. Control Autom.*, vol. 2015-March, no. March, pp. 63–68, 2015.
- [18] Z. Xia, K. Zhou, X. Li, and X. Cui, "A method of robot laser cutting for small holes," in *2016 9th Int. Congr. Image Signal Process. Biomed. Eng. Informatics*, 2016, pp. 1887–1891.
- [19] J. Hatwig, P. Minnerup, M. F. Zaeh, and G. Reinhart, "An automated path planning system for a robot with a laser scanner for remote laser cutting and welding," in *2012 IEEE Int. Conf. Mechatronics Autom.*, 2012, pp. 1323–1328.
- [20] P. Yang, T. Lei, C. Wu, S. Zhao, and J. Hu, "A Fast Calibration of Laser Vision Robotic Welding Systems Using Automatic Path Planning," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–10, 2022.
- [21] Z. Hou, Y. Xu, R. Xiao, and S. Chen, "A teaching-free welding method based on laser visual sensing system in robotic GMAW," *Int. J. Adv. Manuf. Technol.*, vol. 109, no. 5, pp. 1755–1774, jul 2020. [Online]. Available: <https://doi.org/10.1007/s00170-020-05774-0>
- [22] L. Yang, E. Li, T. Long, J. Fan, and Z. Liang, "A Novel 3-D Path Extraction Method for Arc Welding Robot Based on Stereo Structured Light Sensor," *IEEE Sens. J.*, vol. 19, no. 2, pp. 763–773, 2019.
- [23] J. Muhammad, H. Altun, and E. Abo-Serie, "Welding seam profiling techniques based on active vision sensing for intelligent robotic welding," *Int. J. Adv. Manuf. Technol.*, vol. 88, no. 1–4, pp. 127–145, 2017.
- [24] Y. Xu, G. Fang, S. Chen, J. J. Zou, and Z. Ye, "Real-time image processing for vision-based weld seam tracking in robotic GMAW," *Int. J. Adv. Manuf. Technol.*, vol. 73, no. 9–12, pp. 1413–1425, 2014.
- [25] M. Dinham and G. Fang, "Autonomous weld seam identification and localisation using eye-in-hand stereo vision for robotic arc welding," *Robot. Comput. Integr. Manuf.*, vol. 29, no. 5, pp. 288–301, 2013. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0736584513000057>
- [26] Z. Ye, G. Fang, S. Chen, and M. Dinham, "A robust algorithm for weld seam extraction based on prior knowledge of weld seam," *Sens. Rev.*, 2013.
- [27] Q. Wang, Y. Cheng, W. Jiao, M. T. Johnson, and Y. Zhang, "Virtual reality human-robot collaborative welding: A case study of weaving gas tungsten arc welding," *J. Manuf. Process.*, vol. 48, pp. 210–217, 2019. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1526612519303330>
- [28] L. Hou, X. Chen, K. Lan, R. Rasmussen, and J. Roberts, "Volumetric Next Best View by 3D Occupancy Mapping Using Markov Chain Gibbs Sampler for Precise Manufacturing," *IEEE Access*, vol. 7, pp. 121 949–121 960, 2019.
- [29] W. Peng, Y. Wang, Z. Miao, M. Feng, and Y. Tang, "Viewpoints Planning for Active 3-D Reconstruction of Profiled Blades Using Estimated Occupancy Probabilities (EOP)," *IEEE Trans. Ind. Electron.*, vol. 68, no. 5, pp. 4109–4119, 2021.
- [30] S. Kiciroglu, H. Rhodin, S. N. Sinha, M. Salzmann, and P. Fua, "ActiveMoCap: Optimized Viewpoint Selection for Active Human Motion Capture," in *2020 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 100–109.
- [31] N. Palomeras, N. Hurtós, E. Vidal, and M. Carreras, "Autonomous Exploration of Complex Underwater Environments Using a Probabilistic Next-Best-View Planner," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1619–1625, 2019.
- [32] S. Natarajan, G. Brown, and B. Calli, "Aiding Grasp Synthesis for Novel Objects Using Heuristic-Based and Data-Driven Active Vision Methods," *Front. Robot. AI*, vol. 8, p. 208, 2021. [Online]. Available: <https://www.frontiersin.org/article/10.3389/frobt.2021.696587>
- [33] M. Hegedus, K. Gupta, and M. Mehrandezh, "Towards an Integrated Autonomous Data-Driven Grasping System with a Mobile Manipulator," in *2019 Int. Conf. Robot. Autom.*, 2019, pp. 1601–1607.
- [34] D. Stogl, D. Zumkeller, S. E. Navarro, A. Heilig, and B. Hein, "Tracking, reconstruction and grasping of unknown rotationally symmetrical objects from a conveyor belt," in *2017 22nd IEEE Int. Conf. Emerg. Technol. Fact. Autom.*, 2017, pp. 1–8.
- [35] S. Obwald and M. Bennewitz, "GPU-Accelerated Next-Best-View Coverage of Articulated Scenes," in *2018 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2018, pp. 603–610.
- [36] L. Xu, W. Cheng, K. Guo, L. Han, Y. Liu, and L. Fang, "FlyFusion: Realtime Dynamic Scene Reconstruction Using a Flying Depth Camera," *IEEE Trans. Vis. Comput. Graph.*, vol. 27, no. 1, pp. 68–82, 2021.
- [37] R. Border and J. D. Gammell, "Proactive Estimation of Occlusions and Scene Coverage for Planning Next Best Views in an Unstructured Representation," in *2020 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2020, pp. 4219–4226.
- [38] R. Zeng, W. Zhao, and Y.-J. Liu, "PC-NBV: A Point Cloud Based Deep Network for Efficient Next Best View Planning," in *2020 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2020, pp. 7050–7057.
- [39] C. Wu, R. Zeng, J. Pan, C. C. L. Wang, and Y.-J. Liu, "Plant Phenotyping by Deep-Learning-Based Planner for Multi-Robots," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3113–3120, 2019.
- [40] R. Monica and J. Aleotti, "A Probabilistic Next Best View Planner for Depth Cameras Based on Deep Learning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 3529–3536, 2021.
- [41] C. Collander, W. J. Beksi, and M. Huber, "Learning the Next Best View for 3D Point Clouds via Topological Features," in *2021 IEEE Int. Conf. Robot. Autom.*, 2021, pp. 12 207–12 213.
- [42] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *2011 IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 9–13 2011.
- [43] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Work. open source Softw.*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.
- [44] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IEEE Cat. No.04CH37566)*, vol. 3, 2004, pp. 2149–2154.
- [45] D. Coleman, I. Sukan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," 2014.
- [46] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.

- [47] “Intel RealSense D400 Series Product Family,” <https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf>.



James Akl (Graduate Student Member, IEEE) is currently pursuing the Ph.D. degree in Robotics Engineering at Worcester Polytechnic Institute, Worcester, MA, USA, with the Manipulation and Environmental Robotics Laboratory working on robotic metal scrap recycling. He received the B.E. degree in Mechanical Engineering and the Minor degree in Mathematics from the Lebanese American University, Byblos, Lebanon, in 2019.



Fadi Alladkani (Graduate Student Member, IEEE) received the B.E. degree in Mechanical Engineering from the Lebanese American University, Byblos, Lebanon, in 2019. He is currently pursuing the Ph.D. degree in Robotics Engineering at Worcester Polytechnic Institute, Worcester, MA, USA, with the Manipulation and Environmental Robotics Laboratory working on robotic waste sorting and ensemble learning for grasping.



Berk Calli (Member, IEEE) is an Assistant Professor in the Robotics Engineering Department and in the Computer Science Department at Worcester Polytechnic Institute (WPI) where he directs the Manipulation & Environmental Robotics Laboratory. His main research focus is robotic manipulation and its applications in various domains such as waste and metal recycling, assistive technologies, and within-hand manipulation. His lab develops manipulation algorithms by combining techniques from computer vision, control theory, and machine learning. He

also works on facilitating performance quantification methods for robotic manipulation, and is a founder of the Yale-CMU-Berkeley (YCB) benchmarking project, which provides a platform for the robotic manipulation community to develop and share benchmarking protocols. Prior to WPI, he was a postdoctoral researcher at Yale University’s GRAB Lab, where he worked on vision-based dexterous manipulation with underactuated robotic hands. He received the Ph.D. degree at Delft University of Technology in The Netherlands, where he worked on active sensing aimed at increasing success rates of robotic grasping algorithms.