# Automatic Robot Path Planning for Active Visual Inspection on Free-Form Surfaces

Osama Tasneem[1] and Roel Pieters[1]

***Abstract*— Visual inspection is a crucial yet time-consuming task across various industries. Numerous established methods employ machine learning in inspection tasks, necessitating specific training data that includes predefined inspection poses and training images essential for the training of models. The acquisition of such data and their integration into an inspection framework is challenging due to the variety in objects and scenes involved and due to additional bottlenecks caused by the manual collection of training data by humans, thereby hindering the automation of visual inspection across diverse domains. This work proposes a solution for automatic path planning using a single depth camera mounted on a robot manipulator. Point clouds obtained from the depth images are processed and filtered to extract object profiles and transformed to inspection target paths for the robot end-effector. The approach relies on the geometry of the object and generates an inspection path that follows the shape normal to the surface. Depending on the object size and shape, inspection paths can be defined as single or multi-path plans. Results are demonstrated in both simulated and real-world environments, yielding promising inspection paths for objects with varying sizes and shapes. Code and video are open-source available at: *https://github.com/CuriousLad1000/Auto-Path-Planner***

## I. Introduction

Visual inspection is an indispensable part of any industry [1]. From product quality control to the maintenance of machines, visual inspection plays a vital part in improving the overall production yield and reducing the operation and maintenance costs. Due to the complexity of products and machines, the process of visual inspection is generally done by humans, causing downtime and delay in production and, at the same time, exposing the person to hazardous working environments [2]. However, in many cases, manual visual inspection is difficult to achieve, due to issues in accessibility or the high costs involved, creating demand for automated robotic solutions (see Fig. 1). For example, in the paper mill industry, suction rolls play a vital part in the production of paper from pulp. Based on the stage of use, they perform multiple functions including transportation of the sheet of paper, removing water from the paper, and dewatering the felt. This process can result in the blockage of the hundreds of thousands of holes in the suction roll. Current practices involve manual inspection of these rollers, which is time-consuming and ergonomically unfriendly. As the holes are small and narrowly spaced, operators are required to inspect the holes closely with special lighting equipment. Current growth in technologies has led to new possibilities where the task of visual inspection can be automated with sufficient accuracy [3] by utilizing machine learning techniques.

[1]Cognitive Robotics group, Unit of Automation Technology and Mechanical Engineering, Tampere University, 33720, Tampere, Finland; firstname.surname@tuni.fi

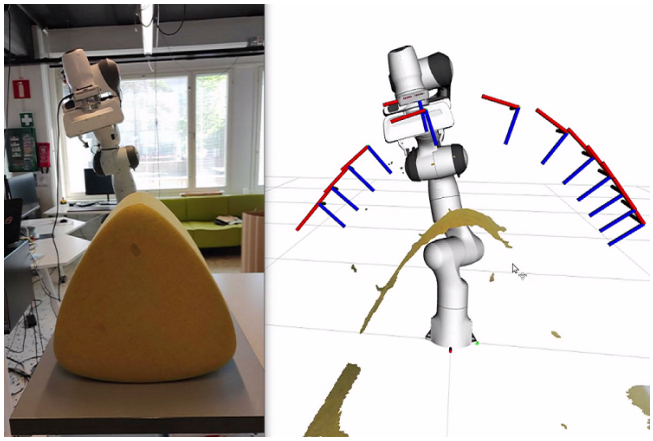


Fig. 1: Automatic path planning utilizes a robot with eye-in-hand camera to extract an inspection path that follows the geometry of the object.

However, the introduction of automation in inspection tasks does not intend to completely replace the human but to reduce the laborious task assigned to humans and improve the overall efficiency and accuracy of the system [1]. Techniques such as deep learning [4], [5], [6] are able to recognize various forms of anomalies that a human inspector used to look for and the implementation of such methods has produced better performance than humans on various applications such as object recognition [7], [8] and sketch search [9]. The accuracy of machine learning techniques being used to detect object anomalies heavily rely on the quality of data for training and validation. Collection of such data can become more difficult due to the size, shape, and location of the object. This is especially true for the inspection of larger infrastructures such as buildings and tunnels. Current data collection methods involve placing multiple cameras at different angles around an object [10] or placing a single camera [11] on top and using an array of mirrors around the object to obtain different views [12]. As products and machines used in factories come in different sizes and shapes, re-arrangement of vision sensors is required to correctly align and gather sample data for training and validation. This introduces another bottleneck in the process to automate visual inspection.

This paper proposes an approach that utilizes a single RGB-D camera, mounted on a robot manipulator to aid in the automation of the inspection process (see Fig. 1).

Visual observations of the geometry of the object and/or the scene then determine the path for the robot to follow to perform visual inspection. Summarizing, the contributions of this work are:

- an approach to filter, process and extract an object profile from depth images.
- object segmentation using clustering to segregate the environment, multiple objects within the field of view.
- the generation of a path plan from the object profile for visual inspection.
- evaluation of the approach with robot and objects of various shapes in simulated and real-world experiments.

The paper is organized as follows. We introduce the paper in Section I. A brief overview of the current developments in path planning is given in Section II, which describes the state of the art and their limitations. The proposed method that extracts an inspection path given a point cloud is described in Section III and its results are reported in Section IV. The results are discussed along with its limitations in Section V. Section VI concludes the work.

## II. Related Work

This section provides an overview of related work in context to robot perception and path planning.

Magnus et al. [13] introduced a methodology in which the operator manually guides the robot along a predefined trajectory, ensuring compliance with joint constraints. The hand-guiding technique might be suitable for motion around objects during the inspection of smaller objects. However, it is not feasible to inspect larger objects using such technique due to time constraints and precision requirements of the task. Lončarević et al. [14] proposed an approach in which they leveraged existing CAD models of the target object to generate inspection trajectories around it. The operator is required to select the correct CAD model with the desired point along the required inspection path. This approach proves advantageous when a CAD model of the target is readily accessible, however, for inspecting unknown objects, an alternative methodology is needed. One approach, as detailed by Roberts et al. [15], involves the utilization of a trajectory optimization model. The authors applied this model to a drone for the purpose of conducting aerial 3D scanning. The method generated drone trajectories to capture and recreate a 'high-fidelity 3D model' of large structures. Another approach presented by Monica et al. [16] centers on Next Best View (NBV) planning, leveraging surfel representations of the environment, in lieu of intricate ray casting operations. The results demonstrate that a score function based on surfels is not only more computationally efficient but also achieves comparable outcomes in terms of reconstruction quality and completeness. Naazare et al. [17] also proposed an online NBV planner for a mobile manipulator robot. The proposed system is designed to facilitate comprehensive exploration as well as user-oriented exploration of an environment, including the inspection of regions of interest. A weighted sum-based information gain function was used to tackle exploration challenges characterized by multiple objectives. While NBV planners excel in robustly exploring unknown environments, it's important to note that these algorithms can be computationally expensive. Conversely, Fan et al. [18] presented an automated solution for 3D object acquisition. Their system implemented adaptive view planning and accommodated objects of varying scales. To devise the optimal path for a 2-axis manipulator, they formulated the path planning algorithm as a Traveling Salesman Problem. The utilization of point clouds as a source of information appears to be a promising and effective approach. However, dense point clouds result in large data volumes and higher processing times. In their work, Arav et al. [19] introduced a context-aware subsampling technique, which effectively preserved the high-resolution details of objects while eliminating data from less critical regions. Research conducted by Yu et al. [20] introduces a point cloud modelling and slicing algorithm designed to address free-form surfaces, specifically for the application of spray painting with a robot manipulator. This algorithm not only preserved the edges within the point cloud but also identified the optimal slicing direction and optimized the movement speed of the spray gun. The spraying trajectory points were based on the cross-section contour points. Wang et al. [21] introduced a path planning algorithm for robotic spray painting that is based on point cloud slicing. The authors employed an adaptive approach to determine the direction of the slice plane, in conjunction with the intersection-projection joint segmentation method, to obtain the points required for constructing the spraying path.

In comparison to the related studies, our approach does not require a CAD model, but generates a path from point cloud data by selecting and combining point clouds with the most data through a majority vote mechanism. The combined point cloud is refined with a hidden point removal process and downsampled to predetermined resolution. Surface normals are then estimated using covariance analysis of Eigen vectors of samples within the local neighborhood of points. Following, a clustering procedure employs DBSCAN clustering [22] to segment and extract valuable objects, after which a visual inspection path is planned.

## III. Methods

This section discusses the overall process in detail, with Alg. 1 and 2 as main methods. Alg. 1 describes the steps from the acquisition of a point cloud to the generation of its profile. Then, Alg. 2 describes the generation of robot target poses from this point cloud profile. Fig. 2 depicts a block diagram that highlights the information flow between various hardware and software blocks used and gives a general graphical overview of the system.

### A. Point cloud acquisition and sampling

Data is collected via a depth camera in the form of a point cloud. To account for noise and enhance robustness, we introduce a sampling filter (step 3 in Alg. 1) that captures multiple frames of color and depth images ($I_{RGB}$, $I_D$) to generate multiple point clouds, as stored in $\mathbf{PCD}_a[]$ array.

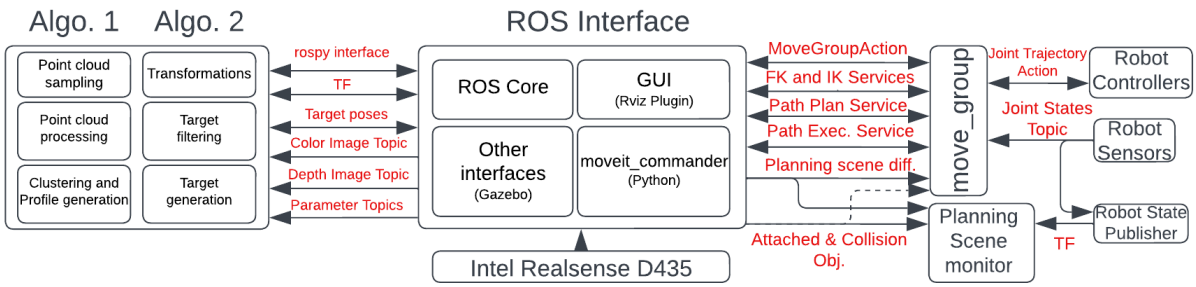



Fig. 2: Block diagram depicting the flow of information between various modules.

**Algorithm 1:** Point cloud processing, clustering, and profile generation.

**Parameters:**
s: number of point clouds to sample
$I_{RGB}$: RGB image frame from camera
$I_D$: Depth image frame from camera
$PCD$: container to hold point cloud
$\mathbf{PCD}_a[]$: array for all sampled PCD
$PCD_p$: Final object PCD profile

**Input** : colour and depth frame, s
**Output** : Filtered point cloud profile $PCD_p$
**STEP 1:** Select initial pose of the robot.
**STEP 2:** Object is in FoV of camera.
**STEP 3:** Sample point clouds.
**foreach** $s$ **do**
  $I_{RGB}$ , $I_D \leftarrow$ grabframe()
  $PCD \leftarrow$ GeneratePointCloud($I_{RGB}$, $I_D$)
  $PCD \leftarrow$ FilterPCD($PCD$)
  $\mathbf{PCD}_a[] \leftarrow PCD$
**STEP 4:** Select point clouds with a majority of similar number of points.
$\mathbf{idx_{list}}[] \leftarrow$ SelectedPointClouds($\mathbf{PCD}_a[]$)
**foreach** $\mathbf{idx_{list}}[]$ **do**
  $PCD \leftarrow PCD$ + $\mathbf{PCD}_a[idx]$
**STEP 5:** Process point cloud and estimate normals.
$PCD \leftarrow$ HiddenPointRemoval($PCD$)
$PCD \leftarrow$ VoxelDownSample($PCD$)
$PCD \leftarrow$ EstimateNormals($PCD$)
**STEP 6:** Generate clusters and object profile.
$Clusters \leftarrow$ ClusterPCD($PCD$)
$PCD \leftarrow$ ClusterSelection($Clusters$)
$PCD_p \leftarrow$ CroppedPCD($PCD$)

**Algorithm 2:** Target generation using filtered profile.

**Parameters:**
$\mathbf{n}_p$: Normalized normals in $PCD_p$
$\mathbf{u}_z$: Unit vector z-axis
$\mathbf{n}_z$: Direction vec. by normal and z-axis
$\theta_z$: Angle between normal and z-axis
$\mathbf{R}_\theta$: Rot. matrix by normals and z-axis
$\mathbf{T}_p$: Poses wrt object for $PCD_p$
$\mathbf{T}_e$: EE poses wrt robot's base

**Input** : Filtered Point cloud profile $PCD_p$
**Output** : EE Targets $\mathbf{T}_e$ wrt to robot's base
**STEP 1:** Generate Rotation matrix
$\mathbf{u}_z \leftarrow [0, 0, 1]$
**foreach** $PCD_p$ **do**
  $\mathbf{n}_p \leftarrow$ normalize($PCD_p.normals$)
  $\mathbf{n}_z \leftarrow$ cross($\mathbf{n}_p$, $\mathbf{u}_z$)/normalize(cross($\mathbf{n}_p$, $\mathbf{u}_z$))
  $\theta_z \leftarrow$ -(acos(dot($\mathbf{n}_p$, $\mathbf{u}_z$)))
  $\mathbf{R}_\theta \leftarrow$ RotMAxisAngle($\theta_z \mathbf{n}_z$)
  $\mathbf{T}_p \leftarrow$ CalcTransforms($\mathbf{R}_\theta$,$PCD_p.points$)
**STEP 2:** Filter world coordinates
$\mathbf{T}_p \leftarrow$ FilterThreshold($\mathbf{T}_p$)
$\mathbf{T}_p \leftarrow$ FilterCloseCoords($\mathbf{T}_p$)
**STEP 3:** Reorder the coordinates
$\mathbf{T}_p \leftarrow$ Reverse($\mathbf{T}_p$)
**STEP 4:** Generate EE targets by transformations
$\mathbf{T}_e \leftarrow$ GenerateEETargets($\mathbf{T}_p$)

These multiple point clouds are then combined to create a better distributed and more dense version of the point cloud data ($PCD$).

### B. Point cloud processing

The acquired point cloud $PCD$ undergoes processing to eliminate undesired elements, including the ground, any visible portions of the robot within the frame, and any other unwanted objects (step 3 in Alg. 1). This filtering procedure is iteratively applied over a predetermined set of sample frames. Subsequently, the optimal frame is identified through a majority vote mechanism, mitigating outliers that deviate from the majority in terms of point count (step 4 in Alg. 1). Background points are eliminated using hidden point removal [23]. Afterward, the point cloud undergoes downsampling to a specified resolution. The importance of downsampling becomes particularly evident when managing dense point clouds, as they can strain computational resources, potentially obscure objects of interest [19], [23] and are unnecessary for our specific application.

### C. Normal estimation

The points within the point cloud contain data about the object's surface in a three-dimensional space relative to the depth camera. In essence, each point can be transformed to obtain its precise position in the world. However, achieving

high-quality data for visual inspection necessitates the camera being positioned orthogonal to the points. These normals are estimated using covariance analysis of Eigen vectors of samples within the local neighborhood of points, a technique described in [24] and implemented through Open3D [23].

### D. Clustering

Due to the distribution characteristics of a captured point cloud, it is essential to cluster and delineate distinct groups of points within the point cloud and isolate an object of interest. It becomes more relevant when there are multiple objects within the Field of View (FoV) of the camera, such as the ground, other objects, sections of robot, etc. Consequently, the DBSCAN clustering algorithm [22] is utilized, which provides the user with a GUI populated with various clusters of which the most useful ones are to be selected. Based on the cluster(s) selected, an object point cloud profile is generated. The users have the flexibility to select the object profile in various patterns and sequences available through the interface including single specific profiles and multi-path profiles generation, which is useful in generating multiple profiles for a better and complete coverage of the object. This profile is then passed on to Alg. 2 to generate targets and stored in $PCD_p$.

### E. Target generation

The process of generating targets is defined by Alg. 2 and takes the object profile $PCD_p$ as input. While the points within the point cloud profile can be used to determine the position of each point, achieving the manipulator's desired pose also requires generating accurate orientations. To achieve this, the algorithm iterates over the points in the profile and computes a rotation matrix $\mathbf{R}_\theta$ (step 1 of Alg. 2) by calculating the axis-angle representation for each point. This rotation matrix is then used to calculate the correct transformations for the object profile $PCD_p$ in world coordinates $\mathbf{T}_p$. Two separate filters are applied to the coordinates (step 2 in Alg. 2) to filter out targets that are beyond user-defined thresholds and anomalous in nature. The first filter removes all coordinates that are too close to the ground or other surfaces and may pose danger to the movement of the end-effector. The second filter calculates the Euclidean distance between each point and if the distance is less than the 2 times the downsample resolution of the point cloud, the coordinate is filtered out. The list of generated targets $\mathbf{T}_p$ is arranged in the correct sequence (step 3 in Alg. 2), and precise transformations are applied to derive the end-effector's world coordinates $\mathbf{T}_e$ for each of these targets (step 4 in Alg. 2).

## IV. Results

### A. Integration

For robot and perception hardware, we utilize the Franka Emika collaborative robotand an Intel Realsense D435 camera, mounted on the end-effector of the robot for eye-in-hand perception. All developments are integrated with ROSwithin Jupyternotebooks using Python. Additionally, Open3D [23] is used for 3D data processing of camera images and MoveIt! for the robot motion planning. The OMPL [25] motion planner provided by MoveIt! was selected as a default planner. The MoveIt python API provides the interface to the task space controller that is primarily used for the robot manipulator's motion. The waypoints are published to the MoveIt! API to make the robot move and record the images and video of the object. Computations are done on a Ubuntu PC with Nvidia GTX 1060 GPU, running ROS Noetic. The Gazebo simulation tool serves as the platform for creating a simulated environment in which the virtual objects are placed and the experiment is conducted. The primary algorithms, along with the graphical user interface (GUI), is implemented using the Python programming language and runs within a Jupyter notebook. For testing, complex shaped objects with different curvatures and size were selected in both simulated and real environments. The objects larger than the robot manipulator were tested in the simulator. These include simulated models of car, tunnel, satellite, submarine, sphere and wall. Smaller models were less than half a meter in length and included physical objects such as bench, aerofoil, 3D printed model of a suction roll, textured metal sheets, inclined planes, etc.

TABLE I: Coverage and planning results (mean of five trials) with 12 profiles and $s = 1$ on three baseline models.

| | Coverage [%] | Time [ms] | Object profile points | Distance [m] [profile×0.01] |
|---|---|---|---|---|
| Car | 93.3 | 596 ± 10.3 | 384 | 3.84 |
| Sphere | 77.3 | 991 ± 9.5 | 489 | 4.89 |
| Shell | 98.6 | 1690 ± 25.9 | 693 | 6.93 |

### B. Point cloud sampling results

To evaluate the effectiveness of the point cloud sampling technique described in Section III-A, we performed the following experiments. We positioned a real inclined metal sheet within the field of view of the depth camera. Adjacent to the metal sheet, we placed a strobe light oriented partially toward the camera. Throughout the experiments, the strobe light operated at a frequency of 3Hz with a 50 percent duty cycle. This setup simulated the varying ambient light intensity conditions affecting both the camera and the surrounding objects. The experiments were divided into three scenarios with varying point cloud samples $s \in \{1, 5, 10\}$, each of which was repeated five times to enhance precision. The selected values here are only to evaluate the efficacy of this algorithm. In actual practise the number of samples will be dependent on ambient noise level and may differ from our test sample.

The resulting point cloud with a single sample $s = 1$ exhibited noticeable gaps in data across the surface and produced sub-optimal targets that did not cover the entire length of the object (see Fig. 3a). In contrast, the combined point cloud was more reliable with sampling set to $s = 5$ and $s = 10$, as shown in Fig. 3b and Fig. 3c respectively.

TABLE II: Point cloud sampling results (mean of five trials) for varying number of samples $s$.

| | Number of points | Sampling time [ms] | Object profile points | Final targets generated |
|---|---|---|---|---|
| $s = 1$ | 715 | 480 | 25 | 8 |
| $s = 5$ | 795 | 1350 | 28 | 11 |
| $s = 10$ | 791 | 2030 | 28 | 10 |

Ultimately, the results, shown in Table II, indicate that a higher number of samples leads to a better distribution of points and a higher density within the point cloud, which improved the total number of final targets generated. A compromise, however, is the increase in sampling time, with more samples included in a point cloud. Determining the ideal number of samples is dependent on current ambient conditions. There can be cases where the filter functions better with ten or more samples.

### C. Point cloud slicing results

Fig. 4 demonstrates the results obtained after applying point cloud processing to a point cloud captured from a simulated object. In Fig. 4a, the raw point cloud is depicted, containing both the primary object of interest and extraneous segments. After filtering and downsizing the point cloud (step 5 in Alg. 1), Fig. 4b is obtained and, following, normal estimation results are depicted in Fig. 4c. The generated object profile (step 6 in Alg. 1) is depicted in Fig. 4d, from which the final targets are generated.

### D. Path planning results

Fig. 5a-c depict the inspection paths and/or targets generated for a variety of simulated objects, including a downscaled 3D model of a car with 28 targets, an aerofoil with 15 targets and a substantial 3D model of a tunnel with 106 targets. Fig. 5d-g depict the inspection paths and/or targets generated using real-world objects, including a bench with 12 targets, an inclined metal sheet with 9 targets, an aerofoil with 13 targets and a model of suction roll with 4 targets.

Targets are generated based on certain physical constraints and can be adapted according to user preferences (see Step 2 in Alg. 2), including the distance of targets to the surface and the spacing in between targets, which, in our experiments, was set to 0.3 meters and 0.02 meters, respectively.

### E. Multi-path planning results

The previous results demonstrated the generation of paths with a single direction, i.e., along a single curvature of the object. Our approach also enables to select the direction for generating the object profile in accordance with a user's preferences. Users can select a specific segment to serve as the object profile or define the number of profiles to completely cover the object's surface. This effectively enables users to create numerous sequences of targets, offering comprehensive coverage of an object's surface, resulting in the generation of multi-path plans, as illustrated in Fig. 6.

### F. Quantitative analysis against baseline models

To evaluate the effectiveness of the proposed model, an experimental setup is established in a simulated environment. The experiment aims to compare the proposed model with three baseline models considered as the ground truth (refer to Fig. 7c). These baseline models were generated from 3D STL files, each containing 5 million points converted into point clouds. For evaluation purposes, the point cloud is voxelized to a standard resolution of 0.01 meters, consistent with our proposed method. The ground truth models consist of 4185 voxels for the car, 7219 voxels for the sphere, and 12724 voxels for the shell, representing variations in size and complexity. The camera mounted on the robot's end-effector maintains a distance of 0.25 meters from the surface during scanning. The surface of the baseline model is scanned, and potential profiles are generated. The program is executed on multiple profiles, and results on surface coverage, planning time, and path length are recorded. Fig. 7a illustrates the relationship between the number of unknown voxels in the region and the number of profiles, while Fig. 7b displays the percentage coverage as the number of profiles increases for the baseline models. These results are also depicted in Fig. 7c, comparing models with two and twelve profiles to the baseline models. Additionally, Table I presents results regarding coverage percentage, distance calculated using the model, and planning time, which includes sampling, filtering, clustering, profile generation, and target generation times.

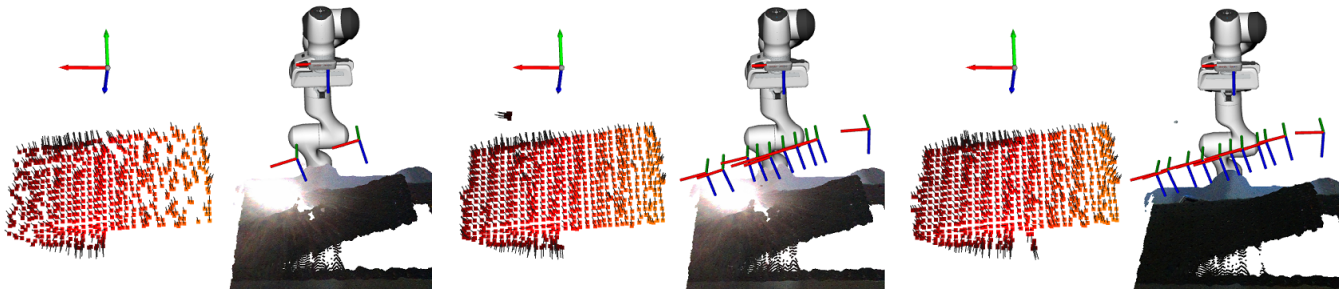
## V. Discussion

As a general result, the path planning approach can be utilized for the visual inspection and 3D modelling of objects without any prior needed information such as object models, training data or human specified trajectories. Depending on the sampling rate of the point cloud, the approach can generate an inspection path in relatively short time with standard perception and computational hardware (see Section IV-A). In addition, our developments are robot-agnostic and are provided open-source to research community.

### A. Simulated vs. real experiments

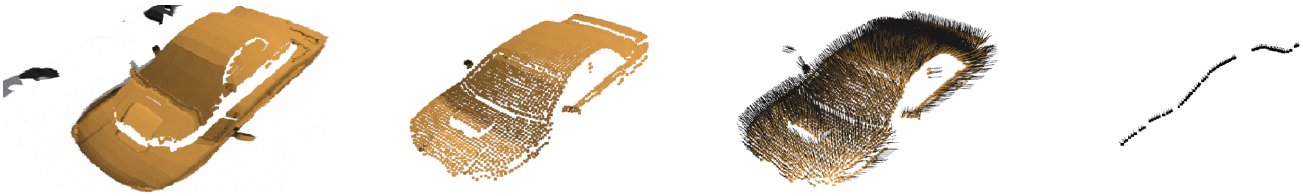
Experiments included both simulated and real-world environments. In simulation, path plans demonstrated better performance due to the absence of physical factors like noise and ambient conditions. In contrast, conducting experiments in the real world proved to be more challenging due to these physical factors. After evaluating the results from both settings, additional filters (step 2 in Alg. 2) were incorporated to enhance the resilience against noise and environmental variations. The integration of these filters resulted in improved outcomes in real-world scenarios, albeit at the expense of increased algorithm complexity and execution time. Nevertheless, the approach offers users the flexibility to configure and optimize the filters to suit the specific characteristics of the object under inspection.

The process of generating targets is influenced by three distinct parameters employed at various program stages. As the targets are derived from the processed point cloud, their



(a) (b) (c)

Fig. 3: Point cloud sampling (step 3 in Alg. 1) results of an inclined plane for $s = 1$ with 629 points (a), $s = 5$ with 792 points (b) and $s = 10$ with 812 points (c). Sampling provides a better distribution of points and a higher density.



(a) (b) (c) (d)

Fig. 4: Simulation results of point cloud filtering and profile generation. (a) depicts the raw point cloud from the depth camera. (b) shows the filtered and down-sampled point cloud (step 3 and 4 in Alg. 1). (c) shows the outcome of the normal estimation process (step 5 in Alg. 1). (d) shows the output generated by the profile generation algorithm (step 6 in Alg. 1).

characteristics are directly influenced by the resolution of the downsampled point cloud. In our experiments, the point cloud downsampling was configured to 0.01 meters, which implies that initial targets cannot be positioned closer than 1 centimeters unless the user further reduces the downsampling scale. The second stage involves the elimination of irregularities in the targets derived from the object profile. It also filters out targets that fall below a user-defined threshold and those that are positioned closer than twice the downsampling scale. Depending on the specific task at hand, this process filters out nearly 50 percent of the total generated targets. The final target filter is a straightforward decimation filter designed to reduce the number of targets by $n$ between two targets. A final observation concerning simulation and real-world experiments relates to motion planning. As a separate motion planning and control system is responsible for determining the robot's trajectory to reach the inspection path targets, the choice of motion planner dictates whether the robot follows an optimal trajectory between these targets or not. Also, motion planning needs to consider the workspace and range of the robot, and avoid singular configurations, which is again affected by the choice of motion planner.

### B. Object profile coverage

The experiment detailed in section IV-F illustrates the number of profiles necessary to cover nearly the entire surface of the object. As depicted in Fig. 7b, it's apparent that the majority of models achieve coverage exceeding 80% with only 6 profiles. However, due to physical limitations and the reach of the robot's end-effector, the program cannot cover the entire surface, leaving certain sections of the models unreachable, notably the surface beneath the object. This results in a skewed outcome, indicating a lower percentage coverage, as seen in the results of the sphere model displayed in Fig. 7. Additionally, Table I outlines the planning time, coverage, profile points, and calculated distance for the three sampled models. The planning time is contingent upon the object's size and the number of profiles employed. Each point in the point cloud being 0.01 meters apart, multiplying it by the number of points in the profile provides an estimate of the end-effector's travel distance over the object's surface.

### C. Comparison to related work

The field of path planning around objects has been thoroughly explored by numerous researchers in diverse contexts. The available approaches can be divided into model-based and model-free methods. Model-based techniques make use of an existing 3D model, whereas non-model-based methods generally rely on online view planning. Some studies concentrate solely on determining the next best view for the purpose of exploring unfamiliar environments, as seen in [17] and [16]. In contrast, others focus on known objects and environments, by requiring CAD models as part of the path planning approach [14]. Our approach relies on online view planning. In our study, path planning is focused on generating inspection targets around unknown objects, without taking into account score functions for determining the next best views. Concerning hardware utilized, certain studies employ drones, as indicated in [15], while others utilize mobile manipulator robots, as demonstrated by [17].

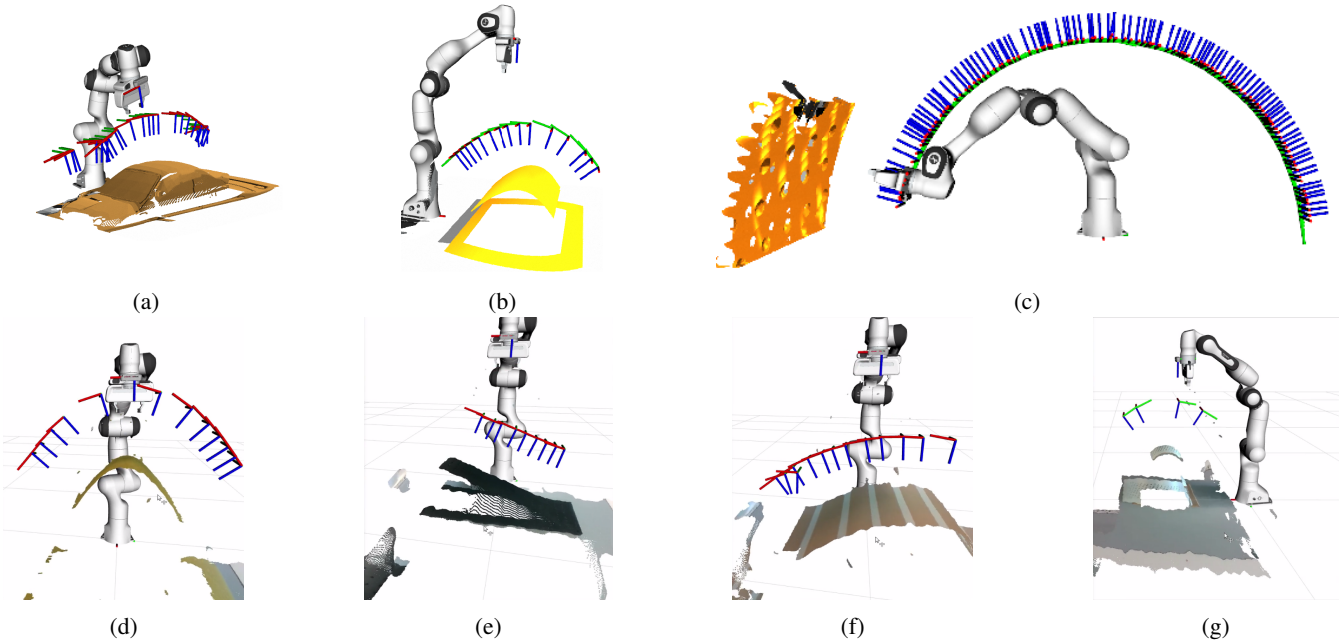



Fig. 5: Results of path targets generated for simulated (a-c) and real (d-g) objects.

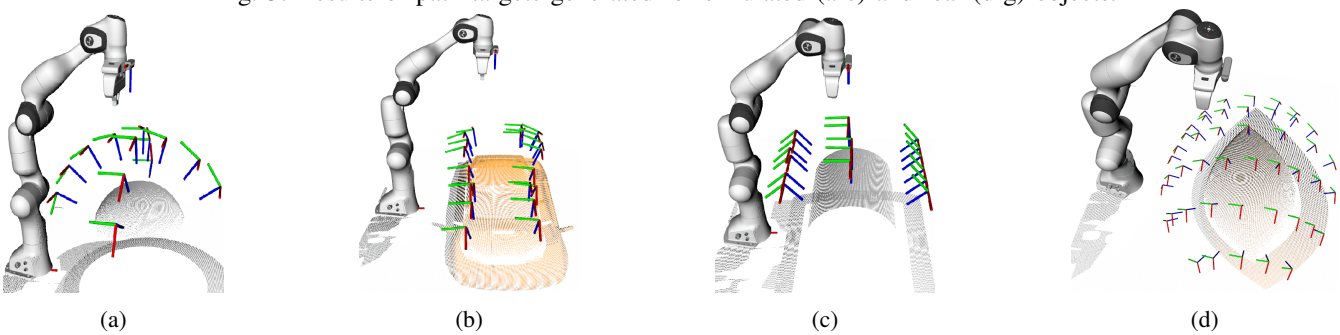



Fig. 6: Results of multi-path targets generated on simulated objects.

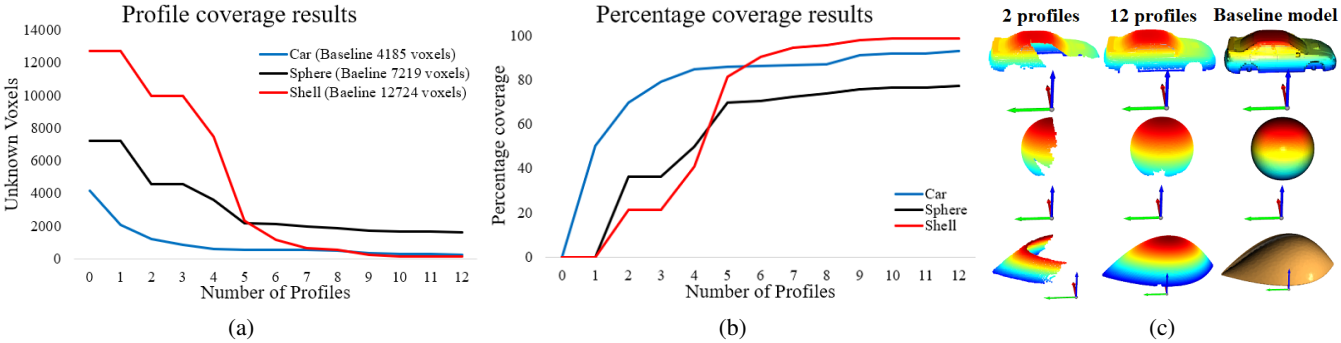



Fig. 7: Object profile coverage results: (a) shows the relationship between unknown voxels and profile count. (b) displays the correlation between coverage percentage and profile count. (c) graphically presents the results for 2, 12 profiles alongside the three models.

Conversely, some focus solely on perception, as shown in [21], for the application of spray painting. In contrast, our research integrates the visual perception approach with a collaborative robot manipulator. It should be noted that our work is robot-agnostic and that any robotic system could utilize the inspection path planning approach.

### D. Limitations

The primary objective in developing our approach was to create a path planner that could be applied universally to objects of varying shapes. While successful results are demonstrated in Section IV, different limitations are observed as well.

One observation during experiments pertained to the introduction of stray points within a point cloud. Given the intricate and dynamic shapes of the objects being inspected, certain cases still exhibited residual points in a point cloud despite using a comprehensive filtering process. For example, in Fig. 4c, the side window pane of the car and points on the left were not entirely filtered, leaving a small cluster of points. This may not pose an issue when generating the object profile from a position within the object's interior, but it could potentially lead to undesired targets when profiling from the object's edges or from a different side.

Furthermore, excessive filtering could have a detrimental effect, causing the profile to lose important details such as curves and edges of the object. In Fig. 4d, there appears to be a minor section absent from the car's roof within the object profile. This absence should not pose significant implications for the inspection task, as long as that part remains within the camera's field of view while moving towards the next target. While our approach offers functionalities to filter targets based on collision thresholds, users are advised to exercise caution and preview generated targets within Rviz before executing any actions.

## VI. CONCLUSION

In this paper, we proposed an automatic path-planning solution for visual inspection, based on the object shape. This addresses the problem of generating a path (position and orientation) that traverses objects of different curvatures and scales, from single or multiple instances of point clouds. The point cloud undergoes a filtering and clustering process to extract the object profile, depending on few constraints (user-defined thresholds, number of point cloud samples, clustering parameters) set by the user. Both single paths and multi-paths can be generated according to the size of the objects to inspect. The approach was evaluated using both simulated and real-world objects of varying sizes and shapes, demonstrating successful path generation and motion execution of a robot with eye-in-hand camera. As a results, the approach can be utilized for the visual inspection and 3D modelling of objects without any prior needed information such as object CAD models or training data. In addition, the approach can run on standard hardware, with single RGB-D camera and any desired robot hardware.

## ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon Europe and Horizon 2020 research and innovation programmes under Grant Agreement no. 101059903 and 871252.

## REFERENCES

[1] J. E. See, C. G. Drury, A. E. Speed, A. Williams, and N. Khalandi, "The role of visual inspection in the 21st century," in *Proc. of the Human Factors and Ergonomics Society Annual Meeting*, vol. 61, 2017, pp. 262 – 266.

[2] S. Agnisarman *et al.*, "A survey of automation-enabled human-in-the-loop systems for infrastructure visual inspection," *Automation in Construction*, vol. 97, pp. 52–76, 2019.

[3] N. Kato, M. Inoue, M. Nishiyama, and Y. Iwai, "Comparing the recognition accuracy of humans and deep learning on a simple visual inspection task," in *5th Asian Conference on Pattern Recognition*, 2020, pp. 184–197.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 770–778.

[5] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 21–37.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *18th International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015, pp. 234–241.

[7] S. F. Dodge and L. Karam, "A study and comparison of human and deep learning recognition performance under visual distortions," in *26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–7.

[8] S. R. Kheradpisheh, M. Ghodrati, M. Ganjtabesh, and T. Masquelier, "Deep networks can resemble human feed-forward vision in invariant object recognition," *Scientific Reports*, vol. 6, p. 32672, 2015.

[9] Q. Yu, Y. Yang, F. Liu, Y.-Z. Song, T. Xiang, and T. M. Hospedales, "Sketch-a-Net: A deep neural network that beats humans," *International Journal of Computer Vision*, vol. 122, pp. 411–425, 2017.

[10] C.-C. Ho, J.-C. Li, T.-H. Kuo, and C. Peng, "Multicamera fusion-based leather defects marking system," *Advances in Mechanical Engineering*, vol. 5, 2013.

[11] Z. Ren, F. Fang, N. Yan, and Y. Wu, "State of the art in defect detection based on machine vision," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 9, pp. 661–691, 2021.

[12] M. A. H. Ali and A. K. Lun, "A cascading fuzzy logic with image processing algorithm–based defect detection for automatic visual inspection of industrial cylindrical object's surface," *Int. Journal of Advanced Manufacturing Technology*, vol. 102, pp. 81–94, 2019.

[13] M. Hanses, R. Behrens, and N. Elkmann, "Hand-guiding robots along predefined geometric paths under hard joint constraints," in *21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016, pp. 1–5.

[14] Z. Loncarevic *et al.*, "Specifying and optimizing robotic motion for visual quality inspection," *Robotics and Computer Integrated Manufacturing*, vol. 72, p. 102200, 2021.

[15] M. Roberts *et al.*, "Submodular trajectory optimization for aerial 3D scanning," in *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5334–5343.

[16] R. Monica and J. Aleotti, "Surfel-based next best view planning," *IEEE Robotics and Automation Letters*, vol. 3, pp. 3324–3331, 2018.

[17] M. Naazare, F. G. Rosas, and D. Schulz, "Online Next-Best-View planner for 3D-exploration and inspection with a mobile manipulator robot," *IEEE Robotics and Automation Letters*, vol. 7, pp. 3779–3786, 2022.

[18] X. Fan, L. Zhang, B. J. Brown, and S. Rusinkiewicz, "Automated view and path planning for scalable multi-object 3D scanning," *ACM Transactions on Graphics TOG*, vol. 35, pp. 1 – 13, 2016.

[19] R. Arav, S. Filin, and N. Pfeifer, "Content-aware point cloud simplification of natural scenes," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–12, 2022.

[20] X. Yu, Z. Cheng, Y. Zhang, and L. Ou, "Point cloud modeling and slicing algorithm for trajectory planning of spray painting robot," *Robotica*, vol. 39, pp. 2246 – 2267, 2021.

[21] G. Wang, J. Cheng, R. Li, and K. Chen, "A new point cloud slicing based path planning algorithm for robotic spray painting," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2015, pp. 1717–1722.

[22] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Knowledge Discovery and Data Mining*, 1996.

[23] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *ArXiv*, vol. abs/1801.09847, 2018.

[24] M. Pauly, "Point primitives for interactive modeling and processing of 3d-geometry," Ph.D. dissertation, ETH Zurich, 2003.

[25] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.