Journal of Manufacturing Processes 134 (2025) 146–157

Contents lists available at ScienceDirect

Journal of Manufacturing Processes

journal homepage: www.elsevier.com/locate/manpro

Full length article

A robotic surface inspection framework and machine-learning based optimal segmentation for aerospace and precision manufacturing

Arun Nandagopal, Jonas Beachy, Colin Acton, Xu Chen [*]

University of Washington, 3900 E Stevens Way NE, Seattle, 98195, WA, USA

ARTICLE INFO

Keywords:
Precision manufacturing
Robotic inspection
Bayesian optimization
Mesh segmentation
Viewpoint generation

ABSTRACT

Quality control is key in the advanced manufacturing of complex parts. Modern precision manufacturing must identify and exclude parts with visual imperfections (e.g., scratches, discolorations, dents, tool marks, etc.) to ensure compliant operation. This inspection process – often manual – is not only time-consuming but also burdensome, subjective, and requires months to years of training, particularly for high-volume production operations. A reliable robotic visual inspection solution, however, has been hindered by the small defect size, intricate part characteristics, and demand for high inspection accuracy. This paper proposes a novel automated inspection path planning framework that addresses these core hurdles through four innovations: camera-parameter-based mesh segmentation, ray-tracing viewpoint placement, robot-agnostic viewpoint planning, and Bayesian optimization for faster segmentation. The effectiveness of the proposed workflow is tested with simulation and experimentation on a robotic inspection of heterogeneous complex geometries.

1. Introduction

In the aerospace industry, meticulous visual inspection is crucial in ensuring flight safety. Even minor flaws such as scratches and dents in critical components like stator vanes can have catastrophic consequences, including airflow imbalances, premature fatigue failure, and potential loss of life [1,2].

As the demand for quality-assured aerospace components skyrockets, inspection becomes a bottleneck hindering industrial throughput. Manual inspection processes are not only costly and time-consuming, but also laborious, subjective, and reliant on extensive training [3]. In fact, human inspectors exhibit an accuracy rate of only around 75% in inspecting precision parts [4].

Several sub-millimeter defect classification methods [5–9] and rob otic image-capturing systems exist, but high-level automation remains largely absent in the aerospace industry due to its high-mix, low-volume (HMLV) manufacturing nature [10]. This necessitates expert reprogramming of inspection systems, which involves manually identifying the locations where the camera needs to be placed using camera-based segmentation of the part and followed by teaching the robot to visit those locations. This process is neither error-proof nor easily reproducible with a new robot or a part.

Therefore to reach rapid manufacturing and quality assurance in aerospace, an agile robotic inspection framework adaptable to diverse part specifications encompassing shape complexity, material variations, and defect detection criteria, is critical. Key features of such a framework include:

- complete surface coverage for diverse geometries,
- occlusion-free image capture,
- optimal number of imaging locations (viewpoints),
- in-focus image acquisition,
- generalizability to different robots, and
- efficient scalability to large parts.

In [11], the authors have developed an adaptive lighting method to generate optimal and even illuminance on complex shapes. We have also built software-hardware integrated robotic solutions for high-impact aerospace parts [12]. Building on these experiences of working with complex geometries, this paper proposes a novel framework that addresses the core features of agile robotic visual inspection through four innovations: camera-parameter-based mesh segmentation, ray-tracing viewpoint placement, robot-agnostic viewpoint planning, and Bayesian Optimization (BO) for efficient segmentation (Fig. 1).

* Corresponding author.
E-mail addresses: arunkru@uw.edu (A. Nandagopal), jbeachy@uw.edu (J. Beachy), actonc@uw.edu (C. Acton), chx@uw.edu (X. Chen).

https://doi.org/10.1016/j.jmapro.2024.12.019
Received 23 May 2024; Received in revised form 28 July 2024; Accepted 9 December 2024
Available online 27 December 2024

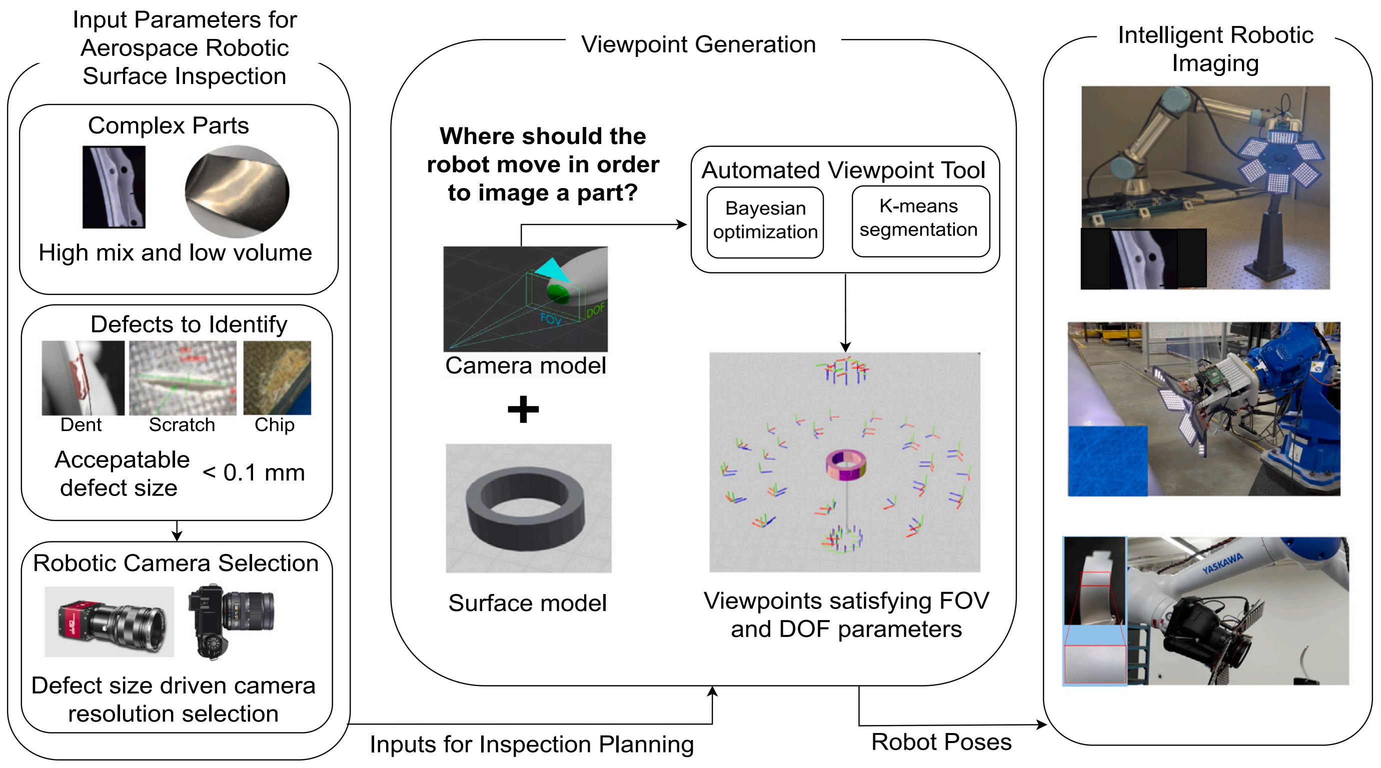


Fig. 1. The proposed workflow for precision visual surface inspection of complex parts. Left: input parameters of the task, including high-mix low-volume complex object geometries, defect characteristics, and camera specifications; middle: viewpoint generation detailing the factors influencing the generation of robot poses for imaging; right: resulting intelligent robotic imaging system comprising a robotic arm equipped with cameras and LED lights for adaptive lighting.

Considering input parameters including part geometries, defect characteristics, and camera specifications, we develop a viewpoint generation detailing the factors influencing the generation of robot poses for imaging. Then, the proposed intelligent robotic imaging system executes the generated viewpoints and paths with advanced lighting and camera controls to capture high-quality images which can be fed in real time to any number of defect detection AI models. Such a generic approach applies to different industrial robot platforms. In this paper, experimentation results are demonstrated using a Universal Robot UR5e collaborative robot manipulator equipped with a customized end-of-arm tool housing a 61MP DSLR camera and 384 independently controlled LEDs, enabling sharp, repeatable, and safe image capture of part surfaces.

The proposed framework capitalizes on two algorithmic contributions of the paper. Our *first* algorithmic contribution relates to the challenge of capturing a part's entire surface from optimal viewpoints in inspection. Manual selection of the viewpoints to collect data is time-consuming and imprecise. On the other hand, existing automated approaches such as identification of flat surfaces using normal vector data of triangles [13] and geometric feature analysis [14], do not address self-occlusion, meaning certain areas of the part cannot be imaged due to obstruction by its own surface. Reinforcement learning methods [15,16] achieve promising results at the cost of an elevated computation cost. In this paper, we propose a novel, efficient, and flexible solution using unsupervised machine learning for viewpoint generation and 3D model segmentation. Surface-based mesh segmentation, which is increasingly gaining traction in inspection [17,18], forms the basis of our approach. Established methods like hierarchical clustering and hybrid region growth [19,20] face challenges due to their neglect of camera constraints during segmentation. This requires a specialized segmentation algorithm that incorporates camera constraints for quantitative assessment. Our proposed methodology integrates unsupervised machine learning through K-means clustering and utilizes the camera's intrinsic model to determine cluster sizes effectively within the Depth of Field (DOF) and Field of View (FOV). This approach is crucial to achieving the level of coverage and precision in imaging required for complete and accurate analysis of a part's surface. It offers the advantage of applying the same algorithm across a range of defect sizes by selecting a camera with spatial resolution approximately one-tenth of the order of magnitude of the defect to be identified. For instance, to detect a 1 mm defect, a camera model with a spatial resolution of 0.1 mm would be suitable.

Inspired by [21], our proposed approach utilizes surface point locations and normal vectors to generate inspection viewpoints, enhancing the precision and effectiveness of defect identification.

Our *second* algorithmic contribution addresses the challenge of systematically determining the number of inspection segments. Methods such as Elbow, Gap statistic, Silhouette Coefficient, and Canopy [22,23] aid in determining K in K-means clustering through iterative evaluation of various metrics. Recent advancements, such as the binning-based silhouette approach and U-K-means algorithm [24,25], enhance search efficiency and eliminate initialization requirements. However, these techniques are tailored for general segmentation problems where the inter-cluster distance is the primary focus. This paper investigates two methods, exponential search, and BO, to find the optimal K value for segmentation based on a quality evaluation index using camera model metrics. BO, in particular, allows for understanding the K-means clustering output and helps predict the right K value with minimal clustering attempts. We present optimal image segmentation with both methods and compare their time performance.

Collectively, the proposed automated inspection framework offers an intelligent algorithm for creating planar patches and generating viewpoints, considering camera constraints. This innovative approach streamlines the inspection process by minimizing human subjectivity, reducing setup time for robotic inspections, and lowering computational costs compared to current methods.

The remainder of the paper is organized as follows. Section 2 provides the methodology of inspection planning from the input of

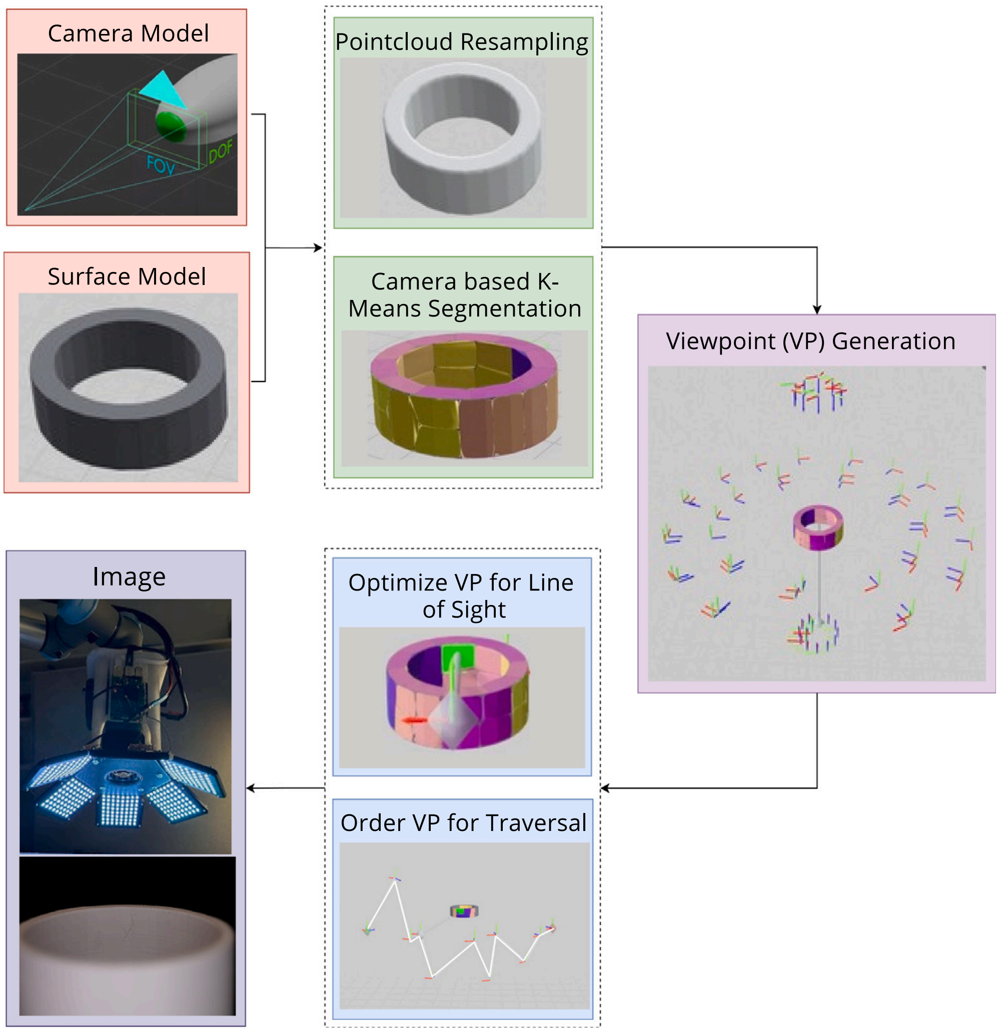

Camera Model
FOV
DOF
Surface Model
Pointcloud Resampling
Camera based K-Means Segmentation
Viewpoint (VP) Generation
Image
Optimize VP for Line of Sight
Order VP for Traversal

Fig. 2. Stages of the proposed automated viewpoint generation framework.

CAD models and camera models to the generation of robot poses. Subsequently, Section 3 evaluates various geometries of aerospace-relevant parts based on segmentation efficiency metrics. Section 4 discusses limits of performance and scope for future work. Section 5 concludes the paper.

2. Methodology

2.1. Overall workflow

The proposed process for calculating viewpoints for imaging a part is delineated in Fig. 2.

The pipeline starts by taking in a surface model in mesh file formats such as STL and OBJ, along with parameters that describe a camera model, such as the camera sensor length and width, magnification, and aperture values. These parameters are used to determine the camera's field of view (FOV) and depth of field (DOF). In Step 2, two procedures are performed. The Pointcloud Resampling procedure transforms the STL model into a point cloud via Poisson-disk sampling. Then, the algorithm described in Section 2.2 is employed to obtain segments of the generated point cloud that fall within the camera's FOV and DOF. This process is referred to as Camera-based K-means Segmentation in Fig. 2.

The number of segments derived is directly related to the camera model described. For example, for a flat surface, the number of segments derived will be directly proportional to the FOV area and does not depend on the DOF value. For a curved surface, the number of segments is again directly proportional to the field of view but indirectly proportional to the DOF. This is because, even though we have enough area in the FOV, we might not be able to fit within the depth of field. This can be correlated to placing the segment of points inside a cylinder, with the circular area representing the field of view and the height of the cylinder representing the depth of field.

Resampling the surface of a mesh file is crucial for three main reasons. First, it helps eliminate symmetry in points for symmetrical objects like spheres or cubes, which can lead to singularity during surface subdivision. Second, resampling addresses issues with the high density of triangular surfaces around vertices, which can affect segmentation by indicating differences in point density. Additionally, flat surfaces in STL file formats often have sparse points, requiring resampling to ensure a uniform density of points for an accurate representation of the CAD model's surface. Lastly, resampling provides control over determining the point density based on the camera's FOV requirements, ensuring there is a minimum number of points needed to form clusters that fit within the field of view. This minimum point density is indirectly proportional to the field of view.

Once the part is divided into segments suitable for imaging, the local geometric properties of each point in the segment are utilized to calculate the position and orientation of the viewpoint in Step 3. This involves computing the average of the normal vectors for each point to obtain the mean normal vector. In addition, the spatial coordinates of each point are averaged to derive the center point for the segments.

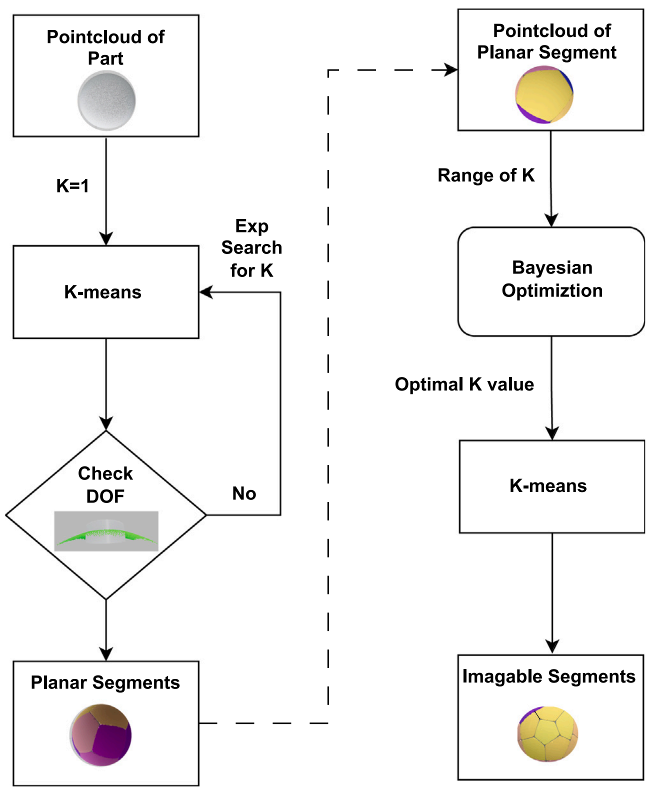


Fig. 3. Workflow of the proposed segmentation algorithm.

Subsequently, the camera position is determined by displacing the center point in the direction of the mean normal by a predefined offset determined by the camera model. The orientation of the principal axis of the camera is then calculated using the opposite direction of the mean normal vector.

Step 4 involves optimizing the viewpoint position and orientation to avoid occlusion by regions of the part itself, as explained in Section 2.4. The results of occlusion avoidance are shown in the final stage of Fig. 2. After occlusion avoidance, these viewpoints are ordered and categorized as explained in Section 2.5 for inspection using a robotic imaging system.

Finally, the robot is commanded to move to these viewpoints to capture in-focus images covering the entire surface of the part. These high-precision images can be fed into the user's choice of AI defect detection model for immediate feedback on manufacturing issues and can be stored locally or remotely for future expert labeling, model training, and traceability.

2.2. Mesh segmentation algorithm

We build upon the common Lloyd's algorithm segmentation technique [26] (commonly referred to as K-means Clustering) and expand to integrate constraints on part geometry, exponential search, and BO. Additionally, we incorporate camera parameters including the FOV and DOF to ensure that the resulting clusters produce well-focused images. Fig. 3 shows the proposed workflow consisting of two segmentation stages. The initial stage involves segmenting the part into planar segments that fall within the camera's DOF. The subsequent stage consists of segmenting these planar segments into regions falling within the FOV.

To perform mesh segmentation using the K-means algorithm, a collection of data points and a method to find an optimal K value are essential. The data points are obtained from the geometric characteristics of the individual points within the point cloud produced during the resampling phase. Specifically, these data points are derived from the coordinates x_i, y_i, z_i, and the unit normals u_i', v_i', w_i' at each point in the point cloud.

Regarding the quantity of segments, determining K for any given point cloud segmentation is challenging. We propose exponential search when the bounds of K are unknown, and BO when such bounds are known.

To derive the planar segments, a feature matrix is constructed by normalizing the coordinates x_i, y_i, and z_i to x_i', y_i', and z_i' such that each coordinate falls within the closed interval $[-1, 1]$, a process known as feature normalization [27]. Eq. (1) shows the process of transformation for the x coordinate. A similar procedure applies to other coordinates.

$$x_i' = -1 + 2\left(\frac{x_i - x_{\min}}{x_{\max} - x_{\min}}\right) \tag{1}$$

Subsequently, a vector $[x_i'\ y_i',\ z_i',\ u_i',\ v_i',\ w_i']$ is constructed for each ith point in the point cloud, and these vectors f_i are stacked to form a p×6 feature matrix ($\mathbf{F_{DOF}}$) as shown in Eq. (2), where p represents the total number of points:

$$\mathbf{F_{DOF}} = \begin{bmatrix} x_1' & y_1' & z_1' & u_1' & v_1' & w_1' \\ x_2' & y_2' & z_2' & u_2' & v_2' & w_2' \\ x_3' & y_3' & z_3' & u_3' & v_3' & w_3' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_p' & y_p' & z_p' & u_p' & v_p' & w_p' \end{bmatrix} \tag{2}$$

The next step involves identifying planar segments by applying the K-means Clustering algorithm [28] to the feature matrix, resulting in clusters of points. Briefly, the algorithm applied to the feature matrix ($\mathbf{F_{DOF}}$) has the following process:

1. **Initialization of K Centroids:**

   $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_k\}$

   where $\mathbf{c}_l$ represents the lth centroid which is initialized by sampling from f_i.
2. **Cluster Assignment:** For each vector $\mathbf{f}_i$ in the feature matrix $\mathbf{F_{DOF}}$, assign it to the nearest centroid:

   $$\mathbf{r}_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_l \|\mathbf{f}_i - \mathbf{c}_l\|^2 \\ 0 & \text{otherwise} \end{cases}$$

   where $\mathbf{r}_{ij}$ is an indicator variable representing the assignment of point $\mathbf{x}_i$ to cluster j.
3. **Centroid Update:** Update each centroid by computing the mean of all points assigned to that centroid:

   $$\mathbf{c}_j = \frac{\sum_{i=1}^{n} \mathbf{r}_{ij}\mathbf{f}_i}{\sum_{i=1}^{n} \mathbf{r}_{ij}}$$

   where $\mathbf{c}_j$ is the new centroid of cluster j.
4. **Convergence Check:** Repeat Steps 2 and 3 until the centroids no longer change significantly or a maximum number of iterations is reached.

At the end of this algorithm, we get points assigned to each of these centroids $\mathbf{c}_j$. These cluster of points represents a collection of points that exhibit geometric similarity, i.e., they are closely positioned and share the same local normal.

In the case of planar segmentation, the bounds of K are uncertain. Therefore, exponential search [29] is preferred. We start with a conservative guess for K, often beginning with 1, and then iteratively adjusting it based on segment evaluation:

1. **Initialization:** Set the initial upper bound as $K = 1$.
2. **Exponential Range Expansion:** Double the upper bound until the segmentation at that value of K produces clusters which satisfy the evaluation function:

   $K = 2^i \quad \text{for } i = 0, 1, 2, \dots$

3. **Binary Search in the Found Range:** Perform a binary search in the subarray defined by the range $[K/2, K]$.

The principle of exponential search allows for a systematic exploration of a wide range of K values. Here, the evaluation function involves verifying whether each point lies within the depth of field of the camera. This involves utilizing the height of bounding box (h_i) of these points in cluster i and comparing with DOF ($h_i < DOF$). Through this iterative process, the optimal value of K is determined to segment the point cloud into planar segments (see Fig. 3: left section).

The next step involves subdividing the planar regions into imageable segments (Fig. 3: right section). This subdivision is achieved by employing the K-means clustering algorithm on the feature matrix consisting of non-normalized coordinate points $[x_i\ y_i\ z_i]$:

$$\mathbf{F_{FOV}} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ \vdots & \vdots & \vdots \\ x_p & y_p & z_p \end{bmatrix} \tag{3}$$

This p by 3 matrix is used to obtain segments of points, where each segment represents a collection of points that are closest in terms of Euclidean distance only. The normal components of individual points are ignored in this feature matrix as the planar segments derived contain points with similar normals.

For the imageable region segmentation, a conservative guess of optimal K value's bounds can be determined. Therefore, BO is preferred, which will be detailed in Section 2.3. The optimal K value determined through this method ensures that all the points are within a radius equal to the least dimension of the camera field of view. As a result, every planar segment is subdivided into image-able sub-regions.

The output of the overall procedure is a set of imageable point clusters. The checks ensure that the cluster's curvature is within the bounds of the depth of field and that its surface area projected onto the imaging plane lies within the camera's FOV.

2.3. Optimal search of imageable point clusters with BO

To efficiently segment a part into the optimal K number of partitions that satisfy the FOV and DOF requirements of the camera, one must minimize the number of evaluations of different K values. This is because it becomes extremely expensive to assess the field of view requirements at large K values. While the exponential search is on the modest time order of $\mathcal{O}[(\log n)]$, the real expense comes from generating the partitions using the K-means algorithm. At a high number of partitions, the computations needed to segment the part are quite intense, and an additional time load will arise from evaluating each partition against the FOV requirements. Thus, at large K values, many partitions, each containing many individual points, get evaluated for each iteration of K. The calculation time scales quickly, making the segmentation of large parts extremely slow.

One way to minimize the number of calls to the K-means algorithm and subsequent FOV evaluations is to quantify how close the segmentation for each K-value is to an optimal segmentation, and then to use this quantification to inform future K selections. This can be accomplished by a cost function that can be evaluated at all K values, designed such that the minimum value of the cost is at the optimal K. If the shape of the cost could be predicted from prior K-value evaluations of the cost function, then future evaluations of the cost function could be focused around the estimated minima. BO is well-suited to optimizing such costly to evaluate, black box cost function. Next, we elaborate the cost function design in Section 2.3.1, and then the setup of BO for segmentation is explained in Section 2.3.2.

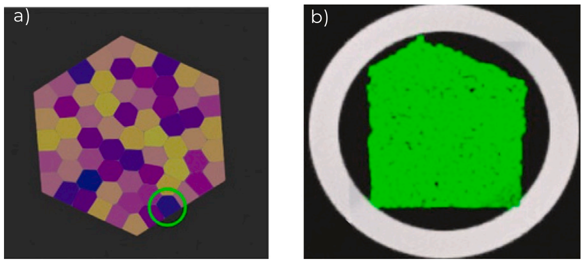


Fig. 4. Segmentation of a planar segment: (a) all the clusters of an example shape satisfying the field of view condition; (b) an individual cluster with all points within the field of view.

2.3.1. Cost function selection

The optimal segmentation occurs when all points within each partition are confined to the camera's field of view (FOV) while minimizing the number of clusters, K. Fig. 4a illustrates an example where a shape is segmented into multiple clusters that satisfy the FOV constraint. Fig. 4b highlights one such cluster, where the FOV is represented by the white circle and the green points indicate the cluster's observed data points. The objective is to design a cost function that is convex, with its minimum corresponding to the optimal K value. By ensuring that the cost function is convex, Bayesian Optimization (BO) can be effectively utilized to find this optimal K by sparsely sampling different cluster partitions.

Bounding-box-based cost function. Consider first a simple-to-evaluate convex cost function using the concept of a bounding box (BB), where the constructed BB around the points provides the length and width of the area occupied by the points. For instance, consider a given FOV with diameter and area ($area_{FOV}$), as shown in Fig. 5a. An ith cluster is depicted in Fig. 5b. We use the area of the green dotted rectangle BB in Fig. 5c for the points that satisfy the FOV condition (green points) as $area(i)_{in}$. Additionally, the area of the black rectangle BB for all points (red and green points) in Fig. 5c is referred to as $area(i)_{out}$. Integrating all the above, the proposed bounding-box cost function is:

$$\text{Cost} = \sum_{i=1}^{n} \left[area(i)_{out} + area_{FOV} - 2 \times area(i)_{in}\right]$$

which penalizes both the area of the region not satisfying the condition and the unoccupied region.

This cost function above performed very well with certain shapes and provides a good starting performance. To evaluate the performance of the cost function, we compute the cost function's minima and the percentage of points not meeting the FOV, hereby referred to as the constraint violation (CV). The cost function for segmentation was tested on common planar shapes with a ratio of the shape's area to the FOV's area equal to 40. The FOV's shape was taken as a circle due to the camera placement requirement. The outcomes were that on planar shapes like hexagon and decagon the CV was 1% and 3% at the predicted minimum point, both fairly good. Whereas for a square and triangle the CV was 4% and 12% at the predicted minimum point, which was not acceptable. The variability is due to certain shapes producing highly angular segments which negatively affect our rectangular cost assignment's accuracy, as shown in Fig. 6, Another problem analyzed with this cost function is that the minimum value for none of the shapes reached a CV of 0%, which was because the area of the unoccupied region in the field of view and area of the region outside the field of view were weighted equally.

Point-based cost function. With the developed understanding of the relationship between shapes and clustering, we now design the full cost function $f(x)$ for implementation. Firstly, we enhanced the cost calculation for points outside the field of view (FOV) by incorporating three parameters: the number of points inside the FOV for the ith

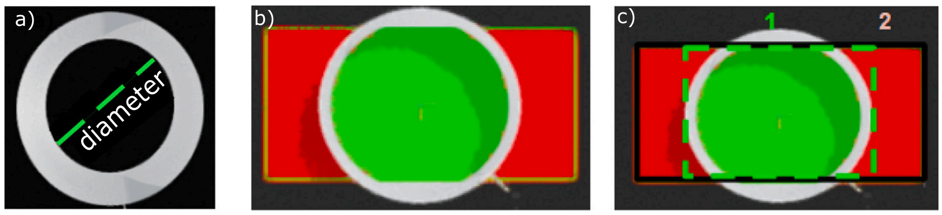


Fig. 5. Elements considered in the cost function calculation for a cluster of points after segmentation: (a) representation of the field of view and the diameter used for cost calculation; (b) points colored green and red based on the field of view condition; (c) two bounding boxes drawn for cost calculation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

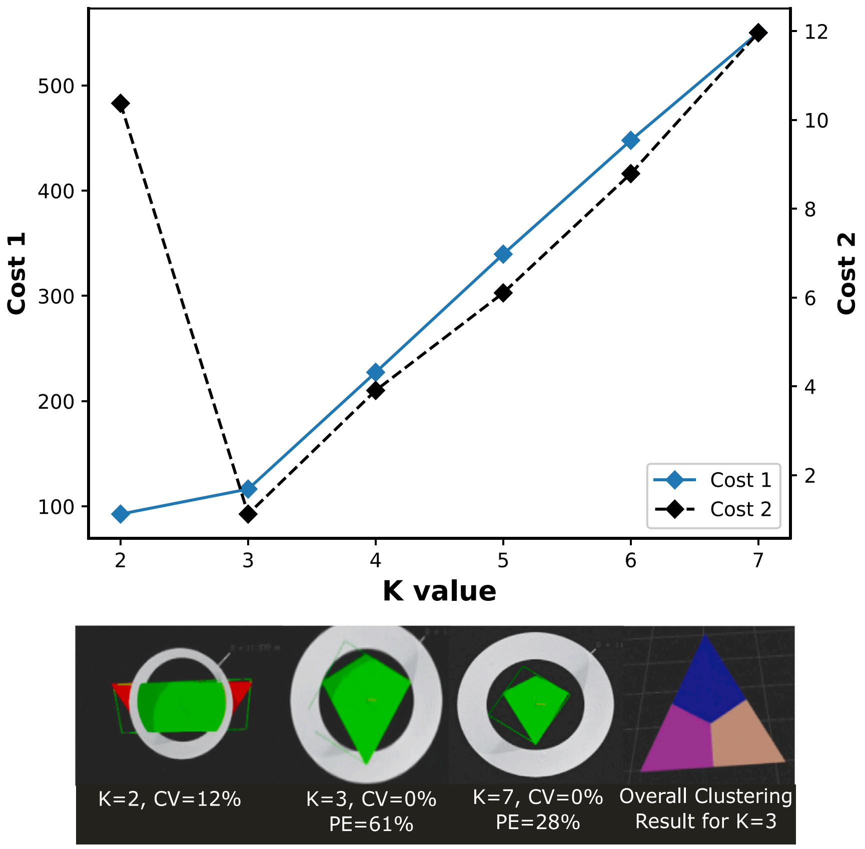


Fig. 6. Comparison of two cost functions for different values of K in K-means segmentation of a triangle-shaped planar patch. The bounding-box-based cost function is represented as cost 1 and the point-based cost function is represented as cost 2.

cluster ($\#points(i)in$), the number of points not meeting the condition ($\#points(i)out$), and n, representing the total number of clusters. The proposed constraint violation percentage (CV) is:

$$CV = \frac{1}{n}\sum_{i=1}^{n}\frac{\#points(i)_{out}}{\#points(i)_{in} + \#points_{out}} \tag{4}$$

The efficiency of points being packed inside FOV is derived from using $Max\#points_{in}$, the max number of points that can be fit within it (calculated using the point density). This term is defined as the packing efficiency (PE) and is calculated via:

$$PE = \frac{1}{n}\sum_{i=1}^{n}\frac{\#points(i)_{in}}{Max\#points_{in}} \tag{5}$$

The second design consideration is to change the equal weighting between CV and PE. This can give more importance to the constraint (CV) in a form such as:

$$f(K) = \lambda \times CV + \phi \times (\frac{1}{PE})^{\beta} \tag{6}$$

where λ, β, and ϕ are positive numbers.

To not have the packing efficiency influence the calculation until the constraint is met, the ϕ term can be set as 0, which makes the cost purely proportional to the constraint violation until the constraint is met. Once the constraint is satisfied, the CV term automatically becomes zero and the packing efficiency comes into effect. To make the first point which attains the CV as 0% we check if any of the points of the cluster lies on the FOV limits and set a value of ϕ zero for that case alone:

$$\phi = \begin{cases} 0, & \text{if CV} \geq 0.001 \\ 1, & \text{if CV} < 0.001 \end{cases} \tag{7}$$

For the specification of circular FOV and polygonal planar shape, we identified λ equal to 100, and the β equal to 2 generated good results. For a triangle, this cost function produced 0% CV at the minimum value and also the highest packing efficiency when satisfying the constraint. Moreover, this cost function worked with similar accuracy on the square, hexagon, and decagon planar segments as well.

2.3.2. Bayesian optimization

With the developed cost function, we now apply BO (see, e.g., [30, 31]), a surrogate-model-based approach to find the optima of the cost $f(K)$. BO is particularly suitable here as our cost function does not have a closed-form solution and is expensive to evaluate. Briefly, BO is an iterative process, using samples of the cost function to calculate a surrogate model, and then selecting the next sample point as the location with the highest probability of finding a new minimum as given by an acquisition function. To perform a BO for the segmentation problem, the surrogate model and type of acquisition function need to be selected from a wide range of options [32]. Additionally, we need to determine parameters that affect the quality of the output, such as the width of the search space and the maximum number of iterations.

The surrogate model utilized for this problem was the Gaussian Process (GP). It was selected due to its popularity to represent non-linear relationships and its success with various use cases [30]. A GP, denoted as

$$f^*(K) \sim \mathrm{GP}(\mu(K), Cov(K, K')) \tag{8}$$

is a function of distributions. For every input value K, it gives the mean and variance of a normal distribution. The input for the GP is the prior mean $\mu(K)$ and covariance $Cov(K, K')$. For the segmentation problem, the prior mean at the start can be taken as zero as the cost function does not have an analytic solution. Furthermore, the covariance function is taken as the standard squared exponential function as the designed cost function is smooth. The equation for the Covariance function for given two arbitrary K values (K_i, K_j) within the search bounds is:

$$Cov(K_i, K_j) = \exp\left(-\frac{\left\| K_i - K_j \right\|^2}{2\ell^2}\right) \tag{9}$$

Here, the hyper-parameter ℓ controls the width of the covariance function, which in turn determines the smoothness of the surrogate model. The ℓ value is tuned based on the magnitude of the cost function search bounds to ensure appropriate smoothness.

As BO explores different values of K for the cost function, the GP surrogate model $f^*(K)$ is updated using Gaussian Process Regression (GPR). When a new K value K_n is sampled and the corresponding cost value $f(K_n)$ is found, the entire GP is updated by taking into account all the K values evaluated, $\mathbf{K_{1:n}} = \{K_1, K_2, \dots, K_n\}$, and the corresponding cost function observations $\mathbf{f_{1:n}} = \{f(K_1), f(K_2), \dots, f(K_n)\}$. The posterior mean and variance that define the updated GP $f^*_{n+1}(K)$ for an arbitrary point K in search bounds are found using Eqs. (10) and (11), respectively:

$$\mu(\mathbf{K}|\mathbf{K_{1:n}}, \mathbf{f_{1:n}}) = \mathbf{Cov_*}^T \mathbf{Cov}^{-1} \mathbf{f_{1:n}} \tag{10}$$

$$\sigma^2(\mathbf{K}|\mathbf{K_{1:n}}, \mathbf{f_{1:n}}) = \mathbf{Cov(K, K)} - \mathbf{Cov_*}^T \mathbf{Cov}^{-1} \mathbf{Cov_*}^T \tag{11}$$

The $\mathbf{Cov_*} = \mathbf{Cov(K, K_{1:n})}$ is the covariance between the test point and the sampled points, and $\mathbf{Cov}$ is the covariance of the sampled points ($\mathbf{K_{1:n}}$). In practice, $\mathbf{K}$ is a vector of evenly spaced points across the whole domain such that the acquisition function has a GP surrogate of the entire cost function. For the derivation and further discussion, readers are referred to [33].

Because the cost function is expensive to evaluate, there is a need to carefully select the next observation point, K_{n+1}, such that the potential information gain is maximized. This is the purpose of an acquisition function, $\alpha(K)$. Crucially, $\alpha(K)$ is calculated using the GP $f^*_{n+1}(K)$. There are many acquisition functions, but they all balance the trade-off between exploration and exploitation. The exploration refers to evaluating areas with high uncertainty and the exploitation for evaluating areas near previously sampled low-cost values. In this work, the celebrated Expected Improvement (EI) [34] function is used. EI at an arbitrary K value is obtained by comparing the minimum value of all the cost functions evaluated in the previous n iterations $f(K_{min})$ and the GP evaluated at $f^*_{n+1}(K)$. The formulation of the EI is:

$$\alpha_n^{EI}(K_i) = \mathbb{E}[\max\{0, f(K_{min}) - f^*_{n+1}(K)\}] \tag{12}$$

The EI accounts for both the likelihood that sampling at K_{n+1} will return a new minimum and the potential magnitude of that improvement in the surrogate model, making it a balanced acquisition function and one well-suited for our needs. After evaluating, the next point is:

$$K_{n+1} = \mathrm{argmax}_K(\alpha_n^{EI}(K_n)) \tag{13}$$

In the formulation of the GP, it is assumed that the cost function can be evaluated at any rational K value in the search bounds. However, the cost function for segmentation is only defined for integer values as the K-means algorithm only works for positive integer inputs. To overcome this, there are three primary methods of adapting the BO to a discrete space as identified in [32]. The naive approach simply rounds the maximum of the acquisition function to the nearest integer before evaluating the cost. This often leads to repeatedly sampling the same point depending on the size of the domain, though research has shown that careful selection of hyperparameters can limit resampling [35]. A second approach is to round inside the covariance matrix. This creates a step-function-like surrogate and acquisition function, which can be difficult to maximize depending on the method. The third method rounds only in the wrapper evaluating the cost (i.e., sample at an integer, but record observation with non-integer value from acquisition). This method is applied in our work. While a simple method, rounding inside the cost wrapper yields impressive results that regularly outperform other discrete surrogate methods [36]. Indeed, while not intended for discrete domains, BO has been applied to a wide variety of discrete situations with minimal effect on performance [37,38]. The complete BO algorithm is shown by Algorithm 1.

The total number of iterations, N, can be set in a variety of ways based on time or computational resources available. In our case, N was determined based on the size of the cost function search bounds.

Algorithm 1 BO for N total cost function evaluations

```
 1: Take $n_0$ initial samples of $f(K)$.
 2: Calculate $f_n^*(K)$ using initial samples.
 3: Set $n = n_0$.
 4: while $n \leq N$ do
 5:     Compute EI acquisition function, $\alpha_n^{EI}(K)$ using $f_n^*(K)$.
 6:     Determine K value using $\mathrm{argmax}_K(\alpha_n^{EI}(K))$.
 7:     Round K value to nearest integer $K_{n_{round}}$ and evaluate $f(K_{n_{round}})$.
 8:     Store K value and $f(K_{n_{round}})$ to all evaluated samples.
 9:     With GPR, calculate $f_{n+1}^*(K)$.
10:     Set $n = n + 1$.
11: end while
12: Return location of minimum value of all cost function evaluation.
```

For the bounds of K, we set the starting point $K_{\min}$ as:

$$K_{\min} = \frac{\text{Length of BB}}{\text{Diameter of FOV}} \tag{14}$$

The rationale behind selecting this $K_{\min}$ value is that it is impossible to have the value of K for the planar segment subdivision less than this number. For the upper bound the K value is set as 3 times the $K_{\min}$ value. Some exceptions to the utilized bounds were when the “length to width” ratio of the bounding box of the shape to be segmented was extremely large.

2.3.3. Usage

The derived cost function is utilized as the function to be minimized using the BO function [39] to find the minimum point location. The bounds for the K values are passed to the BO process and the optimal value of K is derived. This work has been made available online[1].

[1] https://github.com/macs-lab/InspectionMLviewpointGen.git

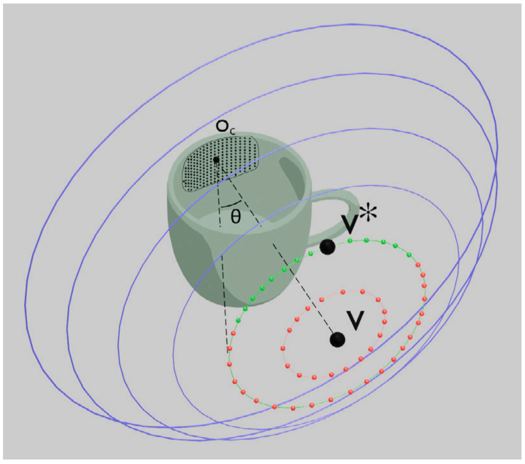


Fig. 7. Illustration of the occlusion avoidance feature with V representing the occluded viewpoint and V* representing the optimized viewpoint.

2.4. Occlusion avoidance

For each formed cluster of points across the surface, we can now form a viewpoint. To do so, we project a point from the center of the cluster by the focal distance of the camera in the direction of the average normal of the set of points. For convex surfaces, the resulting viewpoint will properly capture the desired surface area within the camera's field of view and depth of field; however, projecting viewpoints from concave surfaces can often lead to perspectives occluded by other facets of the part. To bypass these occlusions, a novel method of viewpoint adjustment using ray casting was developed. More specifically, we test each projected viewpoint for occlusion by casting a ray from the viewpoint to each surface point in its corresponding cluster. If the distance traveled by any of the rays does not match the distance from the viewpoint to the surface point within a certain tolerance ϵ, the viewpoint is considered to be occluded, and a new viewpoint must be selected.

For example, in Fig. 7, the origin of the cluster o_c is considered as a pivot point for the original viewpoint, v. New candidate viewpoints are progressively generated and tested within the spherical cap defined by angle $\theta_{\max}$ from the z-axis of o_c. To optimize computational efficiency and bias the choice of optimal viewpoint $v*$ towards v, the search domain is swept by progressively increasing θ from the apex of the cap and testing points sampled uniformly along and in proportion to the circumference of the circle to maintain a uniform distribution of points across the search domain. Once a search of the circle defined by theta results in a set of viable viewpoints, $v*$ is selected from the group to minimize the variance of distances traveled by the rays cast. This ensures that all points in the cluster are as equidistant to the camera as possible, maximizing the likelihood of a well-focused image.

2.5. Viewpoint clustering and traversal

Viewpoints generated for any given part can be categorized into six regions: top, bottom, left, right, front, and back. However, challenges may arise with larger parts, particularly in accessing certain regions, such as the bottom, especially when the part is positioned on a fixture. In such cases, accessing the viewpoints in the bottom region becomes impractical. To overcome this limitation and ensure comprehensive coverage of all regions, a solution involves implementing a rotatable fixture. This fixture allows inspection of one region at a time, for example, imaging the front region initially, followed by rotating the fixture to enable sequential imaging of other facets.

To efficiently facilitate the automatic clustering of viewpoints, this paper utilizes another layer of K-means clustering. With the precise number of regions known (i.e., six regions), the value for K is specified as 6. The input data points for the K-means algorithm comprise the position and normal vectors of the generated viewpoints for the part. This approach enables effective clustering of viewpoints based on their respective regions. Moreover, if interest lies solely in imaging four regions, the value of K can be adjusted accordingly to 4. This methodology enhances the adaptability of imaging these viewpoints without constraints on the robot being used.

For traversing the viewpoints within a particular region, a systematic approach is adopted. The points are sorted from the bottom to the top, ensuring a structured progression. In cases where points possess similar heights from the bottom, the leftmost point is prioritized. This selection criterion facilitates smoother transitions for the robot as it moves from one viewpoint to another within the region. By prioritizing the leftmost point in such scenarios, the traversal process becomes more organized and streamlined, contributing to efficient robotic inspection planning.

The primary objective of the proposed solution in this Section is to ensure that the robot can visit the generated viewpoints to capture images and evaluate image quality and surface coverage when tested across various robots. While viewpoint traversal and trajectory optimization are significant research areas, they are not the central focus of this study.

3. Results

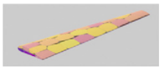
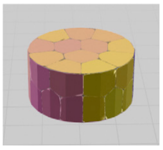
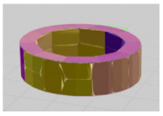
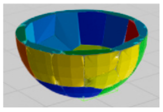
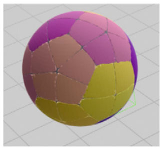
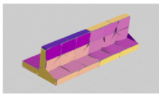
3.1. Mesh segmentation algorithm for different geometries

A qualitative analysis of the segmentation algorithm was conducted to evaluate its performance across various geometries. As this work is one of the first solutions addressing the need for automated flexible robotic path planning with 100% coverage, there were no established open-source benchmarking solutions for this specific problem. Therefore, the geometries selected for evaluating the algorithm were carefully curated to serve as a future benchmark. Additionally, the proposed algorithm[1] is shared in this paper for easier referencing in future research. These geometries were derived using various methods, including TinkerCAD for some shapes and public sources like GrabCAD for others, to ensure comprehensive viewpoint generation testing. The machine used to compute these results is equipped with an AMD Ryzen 9 5900X 12-Core Processor @ 2.80 GHz, a Graphics Card of Nvidia RTX3060Ti, and 32 GB of RAM. To maintain consistency, the surface area of all these meshes was kept within 10% of 2400 square millimeters. Additionally, the area of the image captured was set to 133 square millimeters. These parameters were chosen to minimize runtime while ensuring comprehensive testing across diverse shapes. The approach used for the FOV search for these shapes was exponential search as the number of segments required to divide the planar region was small. The objective was to assess the algorithm's adaptability to different geometries, with the understanding that it could be easily extended to larger parts by adjusting the image capture area accordingly.

The camera parameters remained constant for all parts, with the following values:

- Sensor width = 35.70 mm.
- Sensor height = 23.80 mm.
- Sensor width = 9504 px.
- Sensor height = 6336 px.
- Magnification = 1:3.
- ROI Pixels = 1200 px × 1200 px.
- ROI Area = 11.54 mm × 11.54 mm.
- Depth of field = 4 mm.

Table 1
Qualitative analysis of segmentation algorithm on benchmark geometries.

Index	Shape	Name	$\bar{PE}$	# VP	Time (secs)
a		Wing	0.556	44	11
b		Cylinder	0.54	49	11.5
c		Ring	0.48	55	11.7
d		Bulkhead	0.47	59	14.3
e		Sphere	0.41	60	17
f		T-stiffener	0.35	76	24

- Segment FOV circular diameter = 11.54 mm.
- Spatial resolution = 10 μm.
- Minimum defect size detectable = 100 μm.

The study includes both convex and concave shape categories prevalent in the aerospace industry. Concave shapes, which often combine concave and convex features, were more represented, posing a challenge for the proposed segmentation algorithm. Convex examples are detailed in Tables 1b and 1e, while concave shapes are outlined in Tables 1a, 1c, 1d, and 1f. Notably, Table 1f illustrates a T-stiffener, Table 1a depicts an airfoil-shaped wing, and Table 1d resembles a bulkhead.

These shapes were selected because they represent essential components of aircraft. For instance, the T-stiffener provides structural support, the airfoil-shaped wing generates lift, and the bulkhead partitions the interior space of the aircraft. Other shapes such as the ring, sphere, and cylinder are included for their representative features like flat regions, constant curvature, and hollow regions. These specific shapes were selected to showcase the diverse and crucial functions encountered in the field.

The segmentation process achieved full coverage for all shapes, ensuring every part was successfully segmented. Subsequently, viewpoints were generated from these segments, and occlusion avoidance was performed to adjust for any facets blocking the projected viewpoint. Table 1 presents the detailed results, including the number of generated viewpoints and the time taken. These metrics such as coverage, time taken for viewpoint generation, and the number of viewpoints were captured to provide a framework for future evaluations in this field.

3.2. Efficiency of clustering

The packing efficiency metric in Table 1 offers insight into the segmentation effectiveness for each shape. It is observed that parts with extensive flat surfaces, such as the wing and the cylinder, achieve efficiencies surpassing 0.5. Conversely, shapes with pronounced curvature, such as spheres, rings, and bulkheads, demonstrate packing efficiencies ranging from 0.4 to 0.5. Notably, the T-stiffener case experiences a notable drop in efficiency, primarily due to the formation of planar segments on thin faces that prevent efficient packing of surface points within the circular field of view.

Efficiency serves as a dependable indicator not only of segmentation performance but also of the time required for segmentation and the number of viewpoints required for complete coverage. Parts with lower efficiency demand a greater number of viewpoints for thorough coverage, while those with higher efficiency require fewer viewpoints to achieve equivalent coverage. This correlation highlights the practical importance of efficiency in benchmarking various methods developed in the future.

3.3. Bayesian optimization to find k for different planar segments

Planar segments generally manifest in one of the many common polygons. The efficiency of the BO for the search for minimum viable K is initially tested on a triangle. The cost function generated from a triangle shape with an area of 12662 mm^2 and a camera with a field of view area of 290 mm^2 is shown in Fig. 8. The range space for the BO was set from 1 to 120 (c.f. Section 2.3.2). The parameters of the number of iterations were set to 6. Fig. 8 depicts how the Gaussian Process can predict the minimum K value which is 71 with 6 iterations. Here it uses 3 data points to construct the prior and then uses the acquisition function to predict the minimum value over 3 iterations. This method proved to be faster than an exponential search in which to predict the upper bound (124) would have taken 8 iterations and then a binary search between 64 and 124 to find a K value equal to 71 would have taken 6 iterations. Therefore, 8 iteration searches were reduced due to Bayesian optimization. We can also see that the mean from the Gaussian Process was able to track the real curve accurately.

This method was extended to different shapes of different sizes and it was found to predict the minimum point with a maximum of 9 number of iterations to reach the minimum. An exhaustive list of shapes was evaluated and the number of iterations taken to reach the minimum is shown in Table 2. The BO extended well to other shapes with a very small acceptable tolerance of less than 0.25% constraint violation, which is insignificant when compared to the area these violated points contributed.

To validate the performance of the BO search method with the exponential search the following shapes evaluated previously were re-evaluated using the exponential search method. The number of iterations is compared in Table 2.

From our testing, the BO consistently outperformed the exponential search for the fewest evaluations of the field of view check. The gap between BO and exponential search only widens for larger search ranges. For example, when the K bound is found in between $K = 256$ and $K = 512$ the number of iteration scales to 18(=10+$log(256)$). In the same case when evaluated using the BO, the number of iterations remained equivalent to the case where K was less, by having only 7 iterations to reach the minimum.

3.4. Optimal k value search method time comparison

In the conducted study a sphere was utilized as the test subject to evaluate the performance of exponential search and BO methods in determining the optimal K value. The sphere's area was maintained at 2400 square millimeters, while the area of the FOV varied between 3 and 290 square millimeters, corresponding to the number of clusters ranging from 195 to 1 for each flat region. BO was configured with a fixed number of iterations set to 4, employing a search range spanning from the minimum K value ($K_{\min}$) to three times $K_{\min}$. Conversely, the exponential search initiated its search from $K_{\min}$.

The data obtained from this study is shown in Fig. 9. The analysis revealed a notable performance distinction between the two methods. Notably, the exponential search outperformed BO for K values

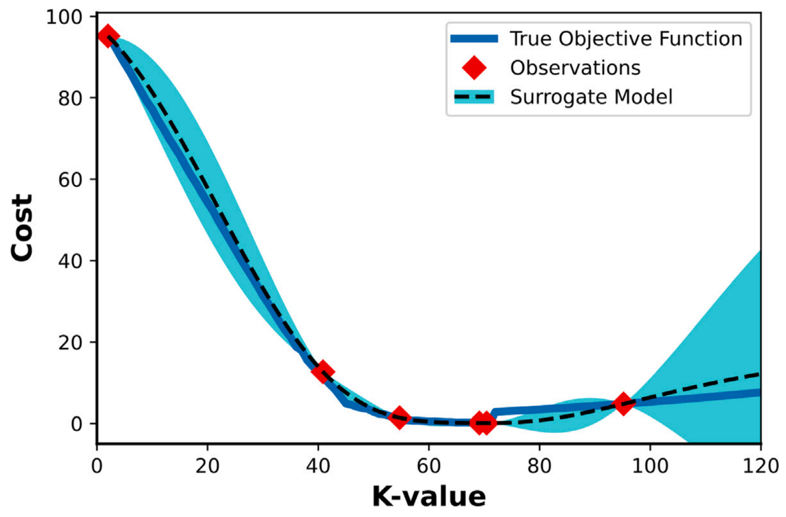


Fig. 8. Predicted cost function surrogate model after 6 K value observation shown against true cost function.

Table 2
Performance evaluation of Bayesian optimization across a variety of test shapes.

Shape	Area of Region (mm^2)	Field of View (mm^2)	CV %	BO Iterations	Exp Search Iterations
Triangle	53 865	290	0.01	5	18
Hexagon	53 865	1161	0.220	8	14
Decagon	11 317	290	0.19	9	13
Square	400	10.7	0.22	8	12

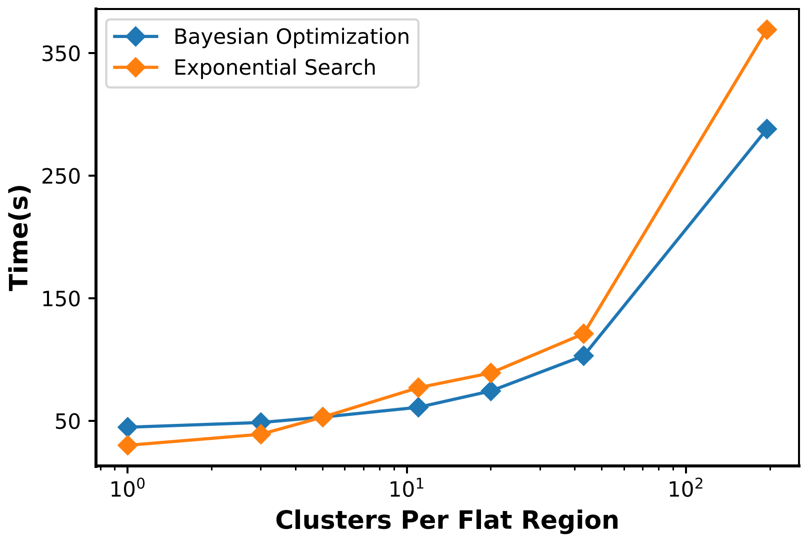


Fig. 9. Time comparison between exponential search and Bayesian optimization for determining the optimal K value on a sphere, with an increasing number of clusters per flat region.

within the range of 1 to 5, primarily due to consistent iteration counts across both approaches. However, beyond this range, BO demonstrated superior efficacy. Particularly, as the optimal K value increased to 195, the time disparity between the two methods approached approximately 100 s. This underscores the increasing efficiency of the BO algorithm with higher K values, rendering it particularly advantageous for identifying K values in larger surface area parts.

3.5. Imaging a part

A 3D-printed hollow cylinder with a diameter of 75 mm and a height of 50 mm served as the test subject for actual viewpoint traversal and inspection. The viewpoints were generated using the same algorithm and then followed by using the proposed robotic imaging system. The part was randomly marked to simulate defects, and after inspection, detailed images of these markings were identified, indicating complete coverage of the part's surface. Fig. 10 showcases the quality and ability to capture defects at various regions of the part.

Fig. 10(a) demonstrates the system's capability to capture defects on the outside surface, while Fig. 10(b) illustrates defect capture on the edge. Fig. 10(c) highlights the system's ability to detect defects on the inside surface. These results collectively affirm the efficacy of the proposed method and underscore its feasibility for deployment.

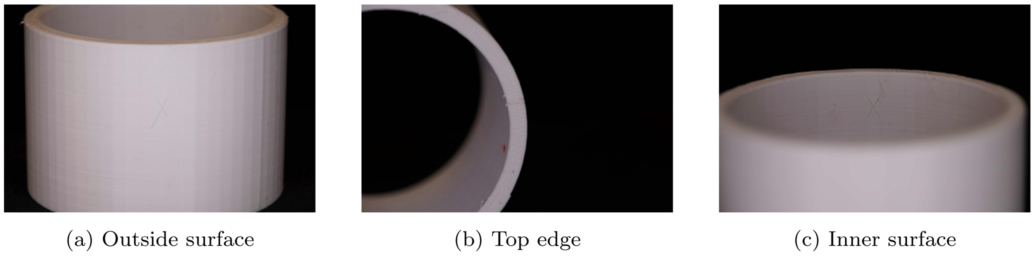


Fig. 10. Images of a hollow cylinder taken using the inspection plan generated from the proposed algorithm. Simulated defects are visible at a spatial precision sufficient for detection by AI.

3.6. Comparison with manual robotic inspection setup

The proposed segmentation algorithm represents a significant advancement in setup time compared to current automated inspection solutions, which typically involve teaching robots to move to specific locations in space based on required images. With traditional methods, setup time increases linearly as the number of imaging regions grows, and segmentation becomes challenging for parts with highly complex surface profiles or large sizes. Additionally, scaling such technology is difficult, as it requires teaching the robot for each new setup, leading to potential errors. However, the segmentation algorithm addresses these challenges by streamlining the inspection process. It eliminates the need for manual teaching, thereby reducing setup time significantly and making segmentation feasible for complex parts. By automating the segmentation process, it enhances scalability and reduces the likelihood of errors associated with manual intervention. Overall, the segmentation algorithm offers a more efficient and reliable solution for part inspection compared to traditional methods.

4. Limitation and future work

4.1. Limitations

Even though the field of mesh segmentation is vast and many solutions exist for different applications, when the general surface inspection problem is considered, it is difficult to arrive at a universal solution. This problem can be compared to the automated spray-painting problem where the goal is to derive waypoints and a path for optimal paint layer covering of any given CAD model [40]. Both these problems are limited by the same factors — there are countless possibilities for requirements of a generic surface inspection, and in all cases, any algorithm should be suitably modified to suit the geometrical constraints posed by the subject under inspection.

With regard to the proposed algorithm, there are a few known limitations. K-means is a non-deterministic algorithm — applying this process to the same object multiple times would result in different segments. This means that it is also possible that the algorithm will settle at a local minimum in terms of the final clusters. To circumvent this, as part of the procedure, this segmentation can be applied to any new part in question until a satisfactory clustering is obtained, and these viewpoints are reused for all subsequent parts of the same geometry. This implies iterating through the segmentation process a couple of times to see if the segmentation efficiency increases drastically, else we are already at the optimal K value.

Another limitation is the ability to deal with small edges as in the case of the T-stiffener, where the packing efficiency falls significantly short because it requires a K value of 23 to arrive at a case where the edges are individual planar patches. This leads to an unnecessarily high number of planar regions to be subdivided.

4.2. Future work

A few aspects of the algorithm have room for improvement. Firstly the nature of repetitive segmentation to avoid local minimum can avoided by using K-means ++ method of initialization which selects initial clusters based on empirical probability distribution which helps in overcoming scenarios with segmentation settling at a local minimum. Secondly, small edges are challenging to inspect. We can improve the capability by using methods such as region-growing algorithms where the part's normals and curvature are used to identify and remove the edges and segment the part into individual faces. Once these faces are determined we can run the existing algorithm on the derived faces which could lead to better segmentation efficiency. Also, modern deep learning techniques could be explored to train a neural network to segment parts that satisfy the camera parameters.

Continued research and development in these areas hold the potential to advance flexible surface inspection methodologies and contribute to more accurate and efficient defect detection processes.

5. Conclusion

This paper presents an agile surface inspection framework that leverages unsupervised machine learning for viewpoint generation and path planning tailored for aerospace-grade components. The proposed approach allows the aerospace industry to deploy a single robotic system for inspecting a variety of parts, significantly reducing the high costs associated with expert robot programming and prototyping. Unlike existing camera and surface model-based methods, our algorithm achieves complete coverage (100%) of complex parts, such as bulkheads, by employing unsupervised machine learning in conjunction with a ray-tracing viewpoint placement strategy. The framework scales effectively with part geometry, demonstrating up to a five-fold speed improvement over traditional exponential search methods due to its integration of Bayesian Optimization.

Validation of the framework was performed using a robotic inspection cell and complex, curved geometries of varying sizes. The results confirm the framework's versatility across a broad spectrum of shapes and its efficiency in detecting defects based on surface models and camera parameters. Future research will focus on enhancing the point cloud pre-processing step through a region growth algorithm to further optimize the packing efficiency for parts with thin edges.

CRediT authorship contribution statement

Arun Nandagopal: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Formal analysis, Data curation, Conceptualization. **Jonas Beachy:** Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis. **Colin Acton:** Software, Resources, Methodology. **Xu Chen:** Writing – review & editing, Writing – original draft, Validation, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declarations

A preliminary version of this work is to be presented at the ASME 2024 International Symposium on Flexible Automation (ISFA).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This project was partially funded by GE Research under project number 401183821, through a subaward of the ARM Institute project TEC-22-01-C-05. The authors would like to thank Shatil Sinha, Matt Hockemeyer, and Anirban Sinha at GE Aerospace (on the manufacturing background), and Sangyoon Back at the University of Washington (robotic integration). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Office of the Secretary of Defense or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] Brauny P, Hammerschmidt M, Malik M. Repair of air–cooled turbine vanes of high–performance aircraft engines–problems and experience. Mater Sci Technol 1985;1(9):719–27.
[2] Advisory Group For Aerospace Research And Development Neuilly-sur-seine (FRAN CE). Erosion, corrosion and foreign object damage effects in gas turbines (Les Consequences de L'Endommagement des Turbines a Gaz Par Erosion, Corrosion et Objets Etrangers). 1994.
[3] Aust J, Pons D. Comparative analysis of human operators and advanced technologies in the visual inspection of aero engine blades. Appl Sci 2022;12(4):2250.
[4] See JE. Visual inspection reliability for precision manufactured parts. Hum Factors 2015;57(8):1427–42. http://dx.doi.org/10.1177/0018720815602389, PMID: 26342002.
[5] Tsai D-M, Jen P-H. Autoencoder-based anomaly detection for surface defect inspection. Adv Eng Inform 2021;48:101272. http://dx.doi.org/10.1016/j.aei.2021.101272.
[6] Gao Y, Li X, Wang XV, Wang L, Gao L. A review on recent advances in vision-based defect recognition towards industrial intelligence. J Manuf Syst 2022;62:753–66. http://dx.doi.org/10.1016/j.jmsy.2021.05.008.
[7] Wang Y, Hong K, Zou J, Peng T, Yang H. A CNN-based visual sorting system with cloud-edge computing for flexible manufacturing systems. IEEE Trans Ind Inf 2020;16:4726–35. http://dx.doi.org/10.1109/TII.2019.2947539.
[8] Eshkevari M, Jahangoshai Rezaee M, Zarinbal M, Izadbakhsh H. Automatic dimensional defect detection for glass vials based on machine vision: A heuristic segmentation method. J Manuf Process 2021;68:973–89. http://dx.doi.org/10.1016/j.jmapro.2021.06.018.
[9] Li W, Li B, Niu S, Wang Z, Wang M, Niu T. LSA-Net: Location and shape attention network for automatic surface defect segmentation. J Manuf Process 2023;99:65–77. http://dx.doi.org/10.1016/j.jmapro.2023.05.001.
[10] Arey D, Gao J, Elsouri M, Le CH. An investigation into the adoption of automation in the aerospace manufacturing industry. 2019, p. 87–92. http://dx.doi.org/10.3233/ATDE190016.
[11] Gerges M, Chen X. Adaptive lighting for curved and nonuniform objects in optomechanical inspection systems. IEEE/ASME Trans Mechatronics 2022;27(6):5792–802. http://dx.doi.org/10.1109/TMECH.2022.3189344.
[12] The ARM Institute. Project highlight: Automated defect inspection of complex metallic parts. 2021, URL https://arminstitute.org/project-highlight-defect-inspection/.
[13] Sheng W, Xi N, Song M, Chen Y, Rankin J. Automated CAD-guided automobile part dimensional inspection. In: Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. symposia proceedings (Cat. No.00CH37065). IEEE; p. 1157–62. http://dx.doi.org/10.1109/ROBOT.2000.844755.
[14] Mosbach D, Gospodnetić P, Rauhut M, Hamann B, Hagen H. Feature-driven viewpoint placement for model-based surface inspection. Mach Vis Appl 2021;32:8. http://dx.doi.org/10.1007/s00138-020-01116-y.
[15] Landgraf C, Meese B, Pabst M, Martius G, Huber MF. A reinforcement learning approach to view planning for automated inspection tasks. Sensors 2021, Vol. 21, Page 2030 2021;21:2030. http://dx.doi.org/10.3390/S21062030.
[16] Wang Y, Peng T, Wang W, Luo M. High-efficient view planning for surface inspection based on parallel deep reinforcement learning. Adv Eng Inform 2023;55:101849. http://dx.doi.org/10.1016/j.aei.2022.101849.
[17] Rodrigues RS, Morgado JF, Gomes AJ. Part-based mesh segmentation: A survey. Comput Graph Forum 2018;37. http://dx.doi.org/10.1111/cgf.13323.
[18] Shamir A. A survey on mesh segmentation techniques. Comput Graph Forum 2008;27. http://dx.doi.org/10.1111/j.1467-8659.2007.01103.x.
[19] Garland M, Willmott A, Heckbert PS. Hierarchical face clustering on polygonal surfaces. In: Proceedings of the symposium on interactive 3d graphics. 2001, http://dx.doi.org/10.1145/364338.364345.
[20] Cohen-Steiner D, Alliez P, Desbrun M. Variational shape approximation. In: ACM SIGGRAPH 2004 papers, SIGGRAPH 2004. 2004, http://dx.doi.org/10.1145/1186562.1015817.
[21] Kaljaca D, Vroegindeweij B, Henten E. Coverage trajectory planning for a bush trimming robot arm. J Field Robotics 2020;37:283–308. http://dx.doi.org/10.1002/rob.21917.
[22] Yuan C, Yang H. Research on K-value selection method of K-means clustering algorithm. J 2019;2(2):226–35. http://dx.doi.org/10.3390/j2020016.
[23] Mehar AM, Matawie K, Maeder A. Determining an optimal value of k in K-means clustering. In: 2013 IEEE international conference on bioinformatics and biomedicine. 2013, p. 51–5. http://dx.doi.org/10.1109/BIBM.2013.6732734.
[24] Punhani A, Faujdar N, Mishra KK, Subramanian M. Binning-based silhouette approach to find the optimal cluster using K-means. IEEE Access 2022;10:115025–32. http://dx.doi.org/10.1109/ACCESS.2022.3215568.
[25] Sinaga KP, Yang M-S. Unsupervised K-means clustering algorithm. IEEE Access 2020;8:80716–27. http://dx.doi.org/10.1109/ACCESS.2020.2988796.
[26] Sushitskii V, Miao HY, Lévesque M, Gosselin FP. Segmentation of peen forming patterns using k-means clustering. J Manuf Process 2024;119:867–77. http://dx.doi.org/10.1016/j.jmapro.2024.04.009.
[27] Shanker M, Hu M, Hung M. Effect of data standardization on neural network training. Omega 1996;24:385–97. http://dx.doi.org/10.1016/0305-0483(96)00010-2.
[28] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in Python. J Mach Learn Res 2011;12:2825–30.
[29] Bentley JL, Yao AC-C. An almost optimal algorithm for unbounded searching. Inform Process Lett 1976;5(3):82–7. http://dx.doi.org/10.1016/0020-0190(76)90071-5.
[30] Brochu E, Cora VM, de Freitas N. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. 2010, arXiv:1012.2599.
[31] Wang J. An intuitive tutorial to Gaussian processes regression. 2022, arXiv:2009.10862.
[32] Garrido-Merchán EC, Hernández-Lobato D. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. Neurocomputing 2020;380:20–35. http://dx.doi.org/10.1016/j.neucom.2019.11.004.
[33] Rasmussen CE, Williams CKI. Gaussian processes for machine learning. The MIT Press; 2005, http://dx.doi.org/10.7551/mitpress/3206.001.0001.
[34] Mockus J, Tiesis V, Zilinskas A. The application of Bayesian methods for seeking the extremum. Towards Global Optim 1978;2(117–129):2.
[35] Luong P, Gupta S, Nguyen D, Rana S, Venkatesh S. Bayesian optimization with discrete variables. In: AI 2019: advances in artificial intelligence: 32nd Australasian joint conference, Adelaide, SA, Australia, December 2–5, 2019, proceedings 32. Springer; 2019, p. 473–84.
[36] Karlsson R, Bliek L, Verwer S, de Weerdt M. Continuous surrogate-based optimization algorithms are well-suited for expensive discrete problems. In: Artificial intelligence and machine learning: 32nd Benelux conference, Leiden the Netherlands, November 19–20, 2020, revised selected papers. Springer International Publishing; 2021, p. 48–63. http://dx.doi.org/10.1007/978-3-030-76640-5_4.
[37] Lizotte D, Wang T, Bowling M, Schuurmans D. Automatic gait optimization with Gaussian process regression. In: Proceedings of the 20th international joint conference on artificial intelligence. 2007, p. 944–9.
[38] Pirot G, Krityakierne T, Ginsbourger D, Renard P. Contaminant source localization via Bayesian global optimization. Hydrol Earth Syst Sci 2019;23(1):351–69. http://dx.doi.org/10.5194/hess-23-351-2019.
[39] Nogueira F. Bayesian Optimization: Open source constrained global optimization tool for Python. 2014, URL https://github.com/fmfn/BayesianOptimization.
[40] Chen H, Fuhlbrigge T, Li X. Automated industrial robot path planning for spray painting process: a review. In: 2008 IEEE international conference on automation science and engineering. IEEE; 2008, p. 522–7.