Towards fully automated inspection of large components with UAVs

offline path planning and view angle dependent optimization strategies

Martin Schörner[1], Raphael Katschinsky[2], Constantin Wanninger[3], Alwin Hoffmann[4], and Wolfgang Reif[5]

University of Augsburg, Augsburg 86159, Germany
{schoerner, katschinsky, wanninger, hoffmann, reif}@isse.de

Abstract. Automation mechanisms are increasingly established in the field of visual quality control. UAVs can be used for particularly large components, such as those used in aircraft or ship production, but also for critical infrastructures. This paper concentrates on the problem of visual quality control in the field of perspective-dependent route planning. It is shown how the requirements for such a system can be implemented and elaborated. Furthermore we investigate how sensor positions can be calculated offline, based on optical- and geometrical requirements and how a trajectory can be planned which contains the found sensor positions for each given area on the component. It is shown how the systems architecture can be designed in order to be able to adapt it to different requirements for the planning of sensor positions and trajectory. The implementation was tested in a simulation environment, evaluated using a benchmark data set and it was shown how above-average results can be achieved on this data set.

Acknowledgements This work was created in collaboration with Kevin Dittel, Teoman Ismail and Christian Adorian from Premium AEROTEC GmbH

1 Introduction

UAVs are increasingly used in the last years in private and commercial as well as in scientific fields due to their easy availability and low cost. This also opens up new possibilities for use in science and industry. With UAVs, camera images can be taken from any perspective and infrastructures can be inspected and measured. The possibilities range from rail tracks, buildings and construction sites [3] to the routine inspection of entire airplanes [17] to the inspection of ship hulls in dry docks [9] A less explored area is the use of UAV in quality assurance in the production of large components such as wind turbine blades and airplane fuselages This work focuses on this area. A problem with many of the projects mentioned is that the inspection is performed manually or it is being only partially automated. The UAV is either controlled manually by a pilot, or a flight route has to be programmed before the flight. Projects with full automation focus more on the complete visual coverage of a component, which is practical, for example, to inspect an entire bridge for cracks. In production, only individual parts of the assembly change from one step to the next, and a complete inspection of the entire component is not necessary. The changes made at the station are known in advance through the construction plan of the assembly. An inspection can therefore focus on the changed positions or newly installed components of the assembly.

In this paper we want to consider the case study of an aircraft fuselage in production. This fuselage is assembled from multiple half-shells. The half-shells are equipped with so-called brackets, which are used for the assembly of further

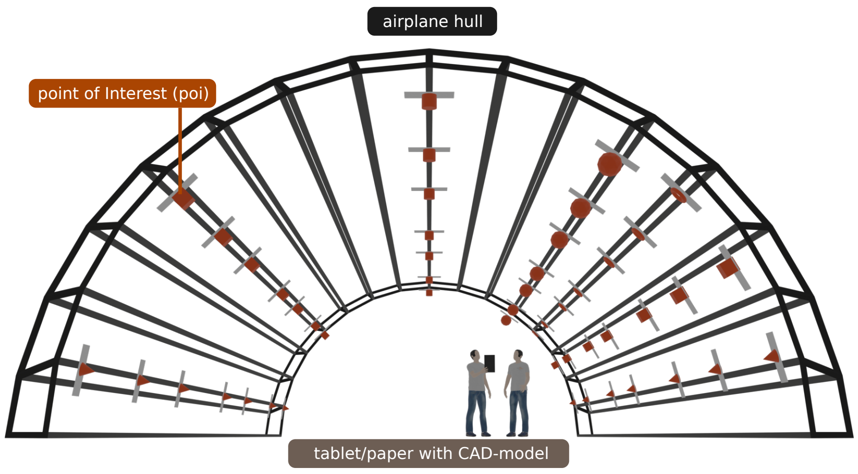


Fig. 1: Inspection of an airplane hull after the installation of various brackets

parts such as cables, insulation or cladding, at several stations during production (see Fig. 1). The position and orientation of the brackets is critical to ensure that subsequent components can be correctly mounted on them. In order to check the correctness of the assembly, a worker checks the newly installed brackets after each work step and compares them with the CAD model of the aircraft using a tablet (see Fig. 1). This process is time consuming. Since the brackets look very similar and the half-shells contain many similar, repetitive sections, human errors can occur easily. To make the process faster and more reliable, approaches for automated error detection already exist [1]. The inspector uses a camera that compares the camera image with the 3D model of the fuselage and detects wrong or incorrectly mounted components. However, this work still requires a worker to inspect the complete half-shell with the camera. The goal of this work is to further automate the inspection and to capture the required camera images using a UAV. This will increase automation and avoid human error. By using UAVs it might even be possible to perform the inspection while the half-shell is on a crane ride from one workstation to the next.

This work presents a concept for the generation of paths for such automated inspections. First of all, it is described how points can be determined from which objects to be inspected can be seen (see Section 3.1). Then a shortest possible path through all these points is planned, which the UAV can follow during the inspection(see Section 3.5). This concept will then be implemented (see Section 4) and evaluated (see Section 5) based on the case study already presented. Inspection points on a half-shell are calculated and the shortest possible route through exactly these points is calculated (see Section 4.2). In addition, an optimization strategy is presented which further reduces the number of inspection points and thus shortens the inspection (see Section 4.2).

2 Related Work

An overview of the current state of the art in inspections using UAVs is given in Jordan et al. [12]. In most cases, a fully automated inspection using drones aims to completely cover the component to be inspected. This procedure is called *Coverage Path Planning* [10], [6]. The approach by Bircher et al. [5] provides for a two-step inspection: First, viewpoints are generated from which the entire surface can be seen by the UAV. For doing that, the surface of the architecture to be inspected is considered as a triangle mesh. For each of the triangles a viewpoint is calculated from which the triangle can be viewed. The generation of viewpoints can be described as *Art Gallery problem* [16], [11]. After that a shortest possible route covering all viewpoints is calculated. This calculation can be modeled as *traveling salesman problem* [13]. At first it is assumed, that there are no obstacles on the way between directly connected viewpoints. If obstacles occur, the RRT* search algorithm (see [14]) is used to plan a collision free route between the two viewpoints. Since in a production scenario the changed areas of an assembly can be defined precisely for each working stage, a complete inspection of the assembly seems not efficient for our approach. Instead of using triangles of the models mesh, only the changed areas of the assembly should be inspected. This is why our approach uses data of the modified or added parts of the last working step, to generate an optimized route inspecting only relevant points of interest. However the calculation of the shortest path between the viewpoints will be modeled as a traveling salesman problem in our approach.

For the inspection of ship hulls Englot et al. [9] also use Coverage Path Planning. Like the previous procedures, this one is also divided into two steps. However, instead of a UAV, an autonomous underwater vehicle is used due to the nautical environment However, the basic principle of off-line planning and inspection points is applicable to our application. First, the *Redundant Roadmap Algorithm* is used to sample random configurations to ensure complete coverage of the hull surface Then an RRT is used to find a route that connects all *goals*. Additionally, a *Local Coverage Algorithm (LCA)* is used to further reduce the length of the route. By this optimization the originally found route with 192 viewpoints and 246m length could be reduced to 169 viewpoints and 157m length, which can be seen in Fig. 2 The previously presented approaches use the concept of viewpoints that guarantee visibility of a certain area of the part that needs to be inspected. It is assumed that the inspection camera can take any orientation. This can be achieved by using a gimbal, for example. This opens up the problem of calculating the right orientation for the inspection camera. The work of Alarcon-Herrera et al. [4] describes the optimal orientation and position of a viewpoint so that the camera looks directly at it. More precisely, the normal vector to the surface of the POI must run along the optical axis of the camera. This means that the POI is always in the center of the camera image, which ensures optimal visibility during the inspection and prevents distortion of the poi. By keeping a so-called *standoff distance* the position of the viewpoint is described exactly. The standoff distance depends on data such as the resolution of the camera, size of the poi, dimensions of the image sensor and the used opening angle of the camera lens. By optimizing the *standoff distance*, the POI occupies an area of the camera image as large as possible without protruding from it. During an inspection it might not always be possible to look directly at a POI due to obstacles or other parts of the assembly that are blocking the view. To counteract this problem, our approach allows for alternative viewpoints that deviate a certain amount from the normal vector of the poi and the optimal distance from it. Malandrakis et al. [15] describe a procedure for performing non destructive testing on aircraft wing panels using a multirotor. The UAV is equipped with a wide-angle camera and a UV lamp. The video recordings of the camera are streamed to a ground control station and evaluated there. Due to the flat structure of a wing, viewpoints are generated in a grid-like formation on a plane (s. Fig. 3). The route consists of flying off this grid line by line.

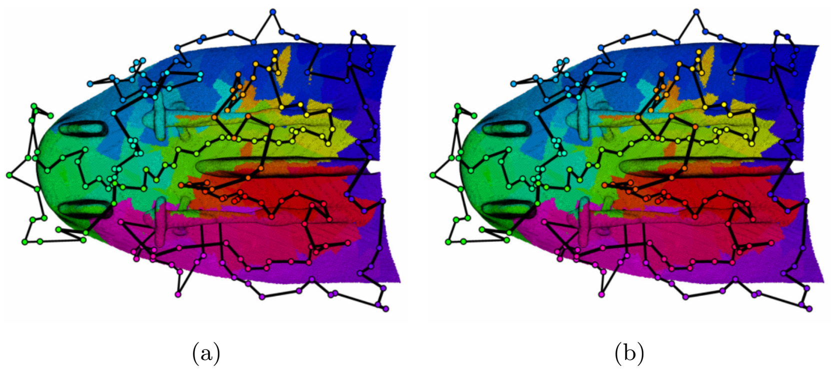


Fig. 2: Optimization of the route length from 192 viewpoints and 246m length (2a) to 169 viewpoints and 157m (2b) [9].

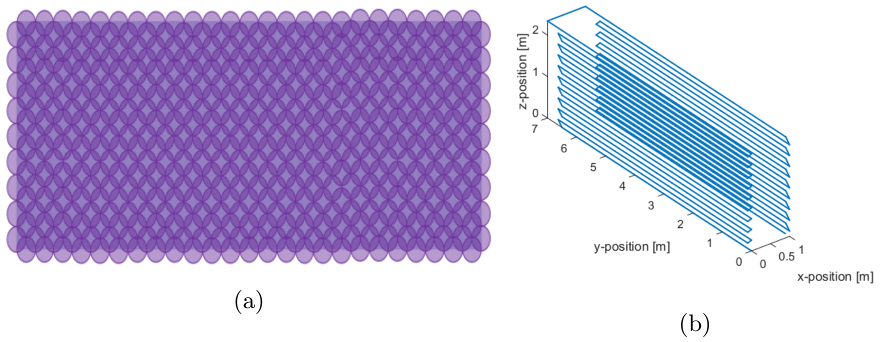


Fig. 3: Grid-like formation of the viewpoints (3a) and the trajectory corresponding (3b) [15].

3 Concept

This chapter covers the conceptual implementation of viewpoint generation and trajectory planning. It is shown how viewpoints can be calculated for POIs of any given orientation and how a trajectory based on the viewpoints can be calculated. The viewpoint planning part shows how viewpoints can be planned so that they meet visual requirements and take into account the characteristics of the camera. The trajectory orientation planning part deals with how the calculation of the trajectory can be interpreted as a traveling salesman problem and solved by ant colony optimization. We also propose a concept for optimizing the length of the resulting trajectory by combining multiple viewpoints to a single one from which multiple POIs can be inspected.

3.1 point of interest

The areas on the component to be inspected during the inspection are called point of interest, or POI for short. These are individual parts or a low-order assembly on the component. POIs are parameterized by their ID, according to the CAD file of the component, their position and orientation, a maximum deviation angle, a semantic annotation and their size. In order to achieve an optimal view of the POI, the deviation angle φ_{max} defines the maximum angle between the optical axis of the camera and the normal of the POI. For the evaluation of the image material, a POI has a semantic annotation, which describes the POI itself in more detail. Finally, the size of the POI is required for the calculation of sensor positions to ensure that the POI is completely visible within the image.

3.2 Viewarea and viewpoint

Based on the aperture angle and the resolution of the camera, a so-called viewarea is calculated for each POI. It contains all possible camera positions that can be taken to inspect a POI and there is exactly one viewarea for each POI. The viewarea is calculated based on the POI size, the maximum deviation angle φ_{max} of the POI and the resolution of the camera. The minimum and maximum distance to the POI is specified by its parameters min-standoff and max-standoff. These specify the radius around the POI in which the POI is completely visible in the picture and the POI can be focused by the camera. The range between the min-standoff-distance and the max-standoff-distance is additionally trimmed by the maximum deviation angle or the opening angle of the camera, depending on which angle is smaller. This can be seen in Fig. 4.

The possible camera positions within the viewarea are so-called viewpoints. A viewpoint is parameterized by its position, list of camera orientations, associated POI and its viewarea. In the best case, the optical axis of the camera and the normal of the POI lie on each other. However, since this is not always possible, the position of the POI (center of the POI) should lie on the optical axis of the camera. From each viewpoint one or more POIs can be inspected, if the viewarea of the respective POIs overlap. Therefore a viewpoint has one or more camera

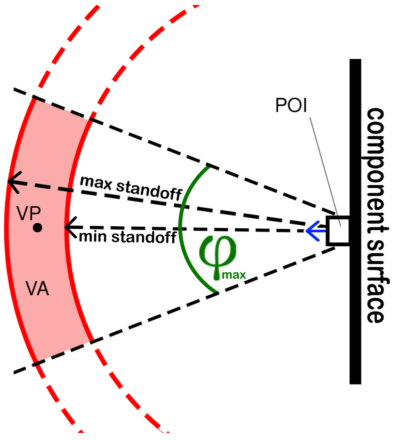


Fig. 4: Viewarea and POI properties.

orientations. Each of these orientations is parameterized as a quaternion and calculated according to Eq. 5 to 1. First the vector $\vec{z}$ is determined by subtracting the x, y and z coordinates of the POI *poi* from the viewpoint *vp* (Eq. 1), to get the line of sight.

$$\vec{z} = [vp_x - poi_x \;\; vp_y - poi_y \;\; vp_z - poi_z] \tag{1}$$

To determine the whole frame of the viewpoint the vectors $\vec{x}$ and $\vec{y}$ are calculated. The vector $\vec{x}$ (Eq. 2) is the cross product of the unit vector $\vec{e_2}$ with the vector $\vec{z}$. Vector $\vec{y}$ (Eq. 3) is the cross product of $\vec{z}$ with $\vec{x}$.

$$\vec{x} = \vec{e_2} \times \vec{z} \tag{2}$$

$$\vec{y} = \vec{z} \times \vec{x} \tag{3}$$

Next, the matrix M is determined from the normalized vectors $\hat{x}$, $\hat{y}$ and $\hat{z}$, which result from the normalization of $\vec{x}$, $\vec{y}$ and $\vec{z}$.

$$M = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} \tag{4}$$

The orientation of the viewpoint is then the quaternion q (Eq. 5).

$$q = q_M \cdot (cos(\frac{\pi}{4}) + 0i + sin(\frac{\pi}{4}) + 0k) \tag{5}$$

where q_M is the quaternion that is formed from the matrix M (Eq. 4. By this calculation the optical axis always points to the position of the respective POI.

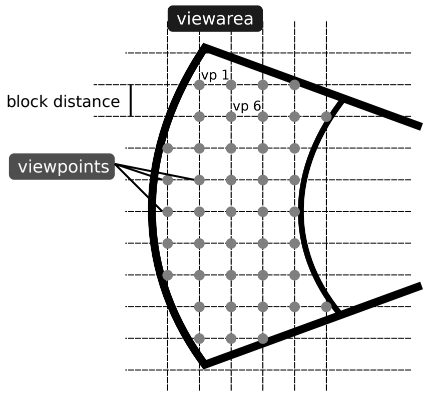


Fig. 5: Viewarea with dividing grid inside. Possible viewpoints lie on the grid.

The benefit of using an area from which the POI can be seen instead of a single point is that alternative viewpoints can be calculated rather easily. If the desired POI can not be seen from a specific viewpoint (e.g. because it is obstructed by an unforeseen obstacle), an alternative viewpoint in the viewarea can be used for the inspection. In order to generate a finite set of alternative viewpoints from the infinite number of possible viewpoints in a viewarea, we sample points in a grid formation with equal spacing within the viewarea. By generating viewpoints that way, we ensure an equal geometric distribution of viewpoints within the viewarea and reduce the number of alternative viewpoints to a manageable amount.

3.3 Viewpoint Optimization

The overall length and duration of the trajectory of the inspecting drone can be shortened by choosing viewpoints, that lie in the intersecting area of multiple viewareas. That way multiple POIs can be seen from a single viewpoint at once. Additionally the drone has to stop at every viewpoint for a specific amount of time in order to give the software time to inspect the POIs. By having less viewpoints the drone needs to make less stops and can therefore complete the inspection in less time. For this purpose it is checked whether overlapping viewareas exist. In the overlapping section of multiple viewareas multiple viewpoints can be inspected from the same position. This leads to fewer sensor positions in total and thus to a shorter trajectory. However, it must be taken into account that an alternative viewpoint in the intersection of two viewareas can have a worse visibility on the POI than the original viewpoint located in the center of the viewarea.

The general Idea is to generate viewareas for each POI and check them for intersections with each other. If Intersecting areas are found, a new viewarea is

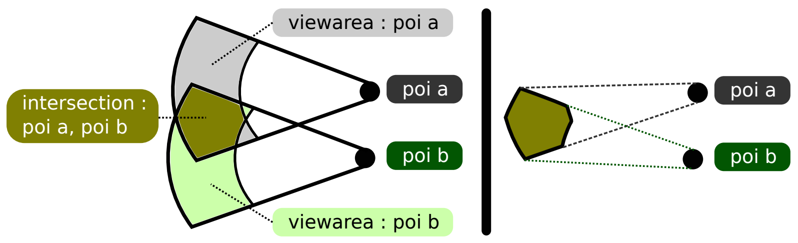


Fig. 6: Combined viewarea from two intersecting viewareas

calculated by using an intersection operation comparable to the intersect operation in Constructive Solid Geometry [18]. The resulting intersecting viewareas are checked for a minimum volume in order to eliminate viewareas that are too small for generating multiple viewpoints. If a combined viewarea meets the minimum Size requirements, its original viewareas are deleted and replaced with the new, combined viewarea.

3.4 Key-Viewpoints

The route for the inspection is a sequence of so-called key-viewpoints. key-viewpoints are those viewpoints that are used as sensor positions for the inspection and hence the trajectory planning. Each POI has at most one key-viewpoint, which implies that the number of key-viewpoints is less than or equal to the number of POIs. If there is no feasible way of reaching any of the points in the viewarea with the drone (because of obstacles or the inability of the drone pointing it's camera at the POI), no key-viewpoint can be found and an error is displayed. Because a UAV has so-called holonomic properties (its controllable degree of freedom with respect to its position corresponds to its total degree of freedom with respect to its position), Viewpoints can be approached directly and the movement between two viewpoints can be described as linear movement.

3.5 Trajectory-Planning

When planning the trajectory, three main requirements must be considered: The trajectory must contain all key-viewpoints, be as short as possible and end at the position where it started. These requirements can be modeled as a Traveling-Salesman-Problem and solved by implementing it using Ant-Colony-Optimization (ACO) [8].
ACO imitates the ants' behaviour when searching for the shortest possible route from the nest to a food source. To find ever shorter routes, an ant leaves a pheromone trail on its route when it has found a food source. The more intense the pheromone trail is the more likely it is that other ants will follow it. As more pheromone can be deposited on a shorter route in the same time span than on a longer route, the intensity of the pheromone increases more on shorter routes

while the pheromone decreases on less frequented routes.
Here the Ant-Colony-System (ACS)[7], a variant of the ACO, was used. ACS differs from the original ACO in three main aspects: i) The state transition rule allows a better balance between exploration of the graph and exploitation of stronger pheromone traces, ii) while constructing a solution, ants use a local pheromone updating rule, iii) only the globally best ant is allowed to deposit pheromone on edges which belong to its tour, via a global pheromone updating rule.
To decide which viewpoint s should be visited next from viewpoint i, the state transition rule is defined by rule given by Eq. 6

$$s = \begin{cases} \underset{c_{ij} \in N(j^p)}{\operatorname{argmax}} \{\tau_{ij} \cdot \eta_{ij}^{\beta}\} & if \;\; q \leq q_0 \\ S & otherwise \end{cases} \tag{6}$$

where $N(j^p)$ is the set of all not yet visited edges.

$$\eta_{ij} = ||i - j||^{-1} \tag{7}$$

where τ_{ij} is the pheromone deposited on edge (i, j), η_{ij} the heuristic information and β a constant that determines the influence of the heuristic information. q is a random value in the interval [0,1] and q_0 is a constant in the interval [0,1] that determines whether the focus should be on exploring the graph or exploiting good pheromone trails. S is a randomly calculated viewpoint s which is not yet visited and is calculated according to Eq. 8.

$$p_{ij}^{k} = \begin{cases} \dfrac{\tau_{ij} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(j^p)} \tau_{il} \cdot \eta_{il}^{\beta}} & if \;\; c_{ij} \in N(j^p) \\ 0 & otherwise \end{cases} \tag{8}$$

The combination of Eq. 6 and 8 is called pseudo-random-proportional rule [7] and favors viewpoints that are connected to the current node with short distances and bigger quantities of pheromone. For updating the pheromone, a local as well as a global update formula is used. While calculating a solution, each ant is visiting and updating its pheromone by Eq. 9

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0 \tag{9}$$

where ρ is the evaporation rate of the pheromone and $\tau_0 = (n \cdot L_{nn})^{-1}$ the initial pheromone level. L_{nn} the route length according to the nearest neighbor heuristic. By using the local update formula already visited edges are getting less interesting for other ants and the exploration of not yet been visited edges increases. When all ants have found a route, only the globally best ant is allowed to update the pheromone on the edges of its route according to Eq. 10

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij}^{best} \tag{10}$$

$$\Delta\tau_{ij}^{best} = \frac{||i - j||_2^{best}}{L_k^{best}} \tag{11}$$

where L_k^{best} is the route length of the shortest route.

4 Implementation

This chapter covers the realization an implementation of the proposed concept. First the underlying software architecture of the offline viewpoint and trajectory planning is presented. Then the implementation and parameter setting of the offline viewpoint and trajectory planning is described.

4.1 Software Architecture

The procedure of the inspection consists of three major parts (see Fig. 7). First viewpoints are being derived from the CAD file and a list of POIs using the Concept of ViewAreas. In this step the ViewArea Optimization is used to merge intersecting ViewAreas. In a next step, a Path is generated for visiting all calculated viewpoints using the ant colony optimization algorithm. In a last step a Trajectory is executed to visit all viewpoints on the Path. This paper will focus on the first two steps of the process. The Trajectory generation will be the focus of future work.

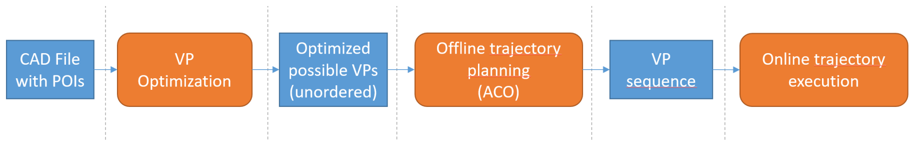

CAD File with POIs
VP Optimization
Optimized possible VPs (unordered)
Offline trajectory planning (ACO)
VP sequence
Online trajectory execution

Fig. 7: Viewarea and POI properties.

In the beginning of the inspection process, an inspector needs to specify relevant POIs in the CAD file. This could be parts, drilled holes or the position of rivets that were added to the product in the last assembly step. In the next step, ViewAreas are calculated for each point of interest based on the Parameters of the inspection camera and maximum deviation angles for the inspection of a POI (see Fig. 8). After that, the optimization of the viewareas is performed by combining intersecting viewareas. This results in a shorter list of viewareas to be visited by the drone. For each viewarea in this List, a discrete number

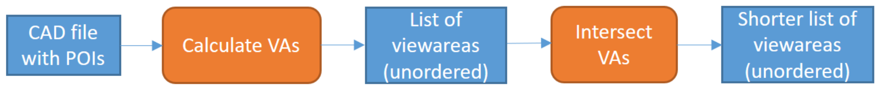

CAD file with POIs
Calculate VAs
List of viewareas (unordered)
Intersect VAs
Shorter list of viewareas (unordered)

Fig. 8: Information flow and change from annotated CAD file to list of optimized ViewAreas

of viewpoints are sampled by picking viewpoints on a grid-like structure inside of the viewarea (see Fig. 9). From these viewpoints, a KeyViewpoint is chosen for each ViewArea. The other viewpoints are stored for later use during the inspection if a POI can not be detected from the perspective of the KeyViewpoint during the trajectory execution.

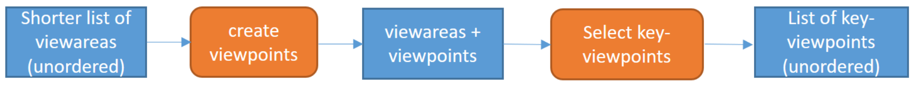

Shorter list of viewareas (unordered)
create viewpoints
viewareas + viewpoints
Select key-viewpoints
List of key-viewpoints (unordered)

Fig. 9: List of KeyViewpoints from list of optimized ViewAreas

In a last step a short path visiting all KeyViewpoints is generated from the resulting list of KeyViewpoints using the ant colony optimization algorithm. This will be used in future work to calculate a trajectory that visits all key-viewpoints. Additionally other factors like static and dynamic obstacles in the drones flightpath and obstacles obstructing the view of the inspection camera on the drone need to be considered in this step.

The requirements for the inspection of components can be very different. For example, an inspection path may be required for more complex components where the structure of it must be considered during the offline planning. It can also be necessary that static external obstacles have to be co nsidered during planning, because they partly overshadow the component or block areas within the component. Therefore, offline planning should allow to offer different planning options and optimization possibilities which can be used depending on the requirements. Independent of the inspection route, it should also be possible to integrate new requirements for the composition and calculation of the viewareas, viewpoints and the trajectory into the system. In order to make the system scalable, to make individual components interchangeable and to be able to adapt the planning to different requirements, a combination of Strategy and Abstract Factory Pattern was implemented for viewarea, viewpoint and Trajectory Planning.
The architectural implementation of the system can be seen in Fig. 14. There you can see that the system consists of the four components Path Strategy, Path Factory, State Space and ACO. The Path Strategy component is responsible for aggregating the data required for viewarea and viewpoint planning. The CAD file, the camera parameters and the desired strategy are transferred to the PathConfigurator and forwarded to the PathController. In a first step the PathController extracts the POIs from the CAD file and in a second step it selects the PathFactory according to the chosen strategy. The PathFactory component receives the extracted POIs and the camera parameters as input and is again divided into two steps. In the first step the viewpointPlanner calculates

and optimizes the viewareas and viewpoints according to the selected strategy. The viewpointPlanner returns a list with all key-viewpoints. The result of the first step is then transferred from the PathFactory to the PathPlanner, which calculates the trajectory according to the chosen strategy in a second step. In Future Work this component also takes care of optimizing the trajectory. For the planning of the trajectory the key-viewpoints are passed on to a trajectory planning component, in this case the ACO component. This component takes over the calculation of the trajectory by the Ant-Colony-System described in section 3.5. As a result, the PathFactory then returns the viewareas and the trajectory. The trajectory can then be executed and the viewareas can be accessed during the inspection if alternative viewpoints are required.

In the remaining part of this section we will focus on the implementation of the viewarea Optimization and the optimization of the length of the path for visiting all key-viewpoints.

4.2 Viewpoint-Planning

The positions and rotations of the points of interest (POI) can be derived from the CAD files. Based on these transformations and including the camera parameters the viewareas are created. The viewareas contain viewpoints, i.e. frames (position and orientation) from which the POI can be seen. Since some viewareas (see Fig. 10) overlap there are viewpoints from which several pois can be seen. This property is used to minimize the route in the number of existing nodes and to speed up both planning and execution. If a POI cannot be seen from a viewpoint during execution, viewareas should be used to generate new viewpoints. The algorithms for the creation of viewareas, as well as the optimization for overlapping viewareas and the distribution of pois within viewareas were written in python 3.x using the blender 2.9x libraries.

Viewareas

From the CAD data the id, its frame and a simple 2 dimensional bounding box of the POI must be extracted. The frame is defined in the center of the bounding box (see Fig. 11). As port for the definition of the POI bounding boxes json is used. The camera parameters like the maximum and minimum distance and the opening angle are defined in a separate json file.

The algorithm uses the minimum and maximum distance to create two spheres and a trapezoid that is extruded from the POI bounding box using the alpha angle to the maximum distance. The three meshes (geometric definition based on triangles) are intersected and transformed to the position and orientation of the bounding box in the last step of the algorithm. In this procedure viewareas are created in which the pois can be seen. viewareas which can see the complete bounding box in every point of view are much smaller and are explicitly not desired for later evaluations of the optimization at runtime in this project.

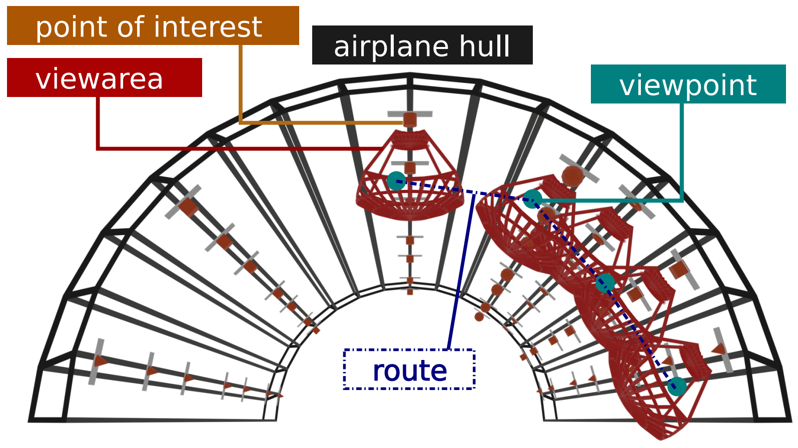


Fig. 10: airplane hull with some point of interests, viewareas, point of views and a simple optimized route

Viewarea intersection

Especially in the example of the inspection of aircraft inner paneling, the pois are very close together. So that not every POI gets its own viewpoint the viewareas should be checked for overlaps. The algorithm checks for any possible overlap and decides on the basis of a heuristic whether the two viewareas in question will be intersected. The heuristic should prevent that too small areas are created (as shown in Fig. 12, in which the viewareas VA 1 and VA 2 were not intesected due to heuristics).

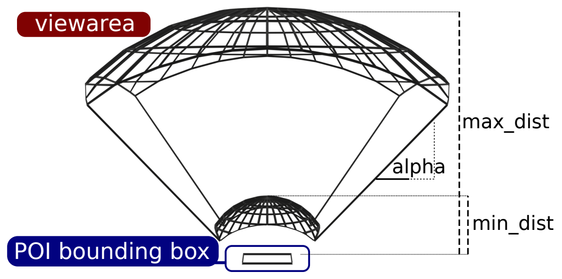


Fig. 11: Viewpoint, generated from the dimensions of the POI bounding box and the camera parameters: minimum distance (min_dist), maximum distance (max_dist) and opening angle (alpha) of the camera

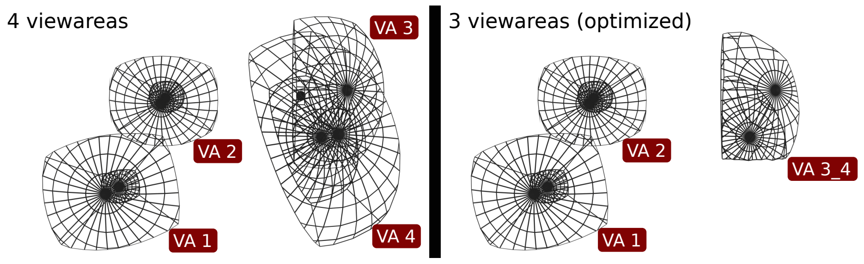


Fig. 12: On the left side there are four viewareas, where VA 1 and VA 2 as well as VA 3 and VA 4 overlap. On the right side of the picture, only 3 viewareas can be seen, with VA 3 and VA 4 being combined

The parameters for the heuristic are the number of desired viewpoints and their block spacing in 3 dimensional space. These parameters are used to estimate the extent of an intersection between two viewareas. After a successful intersection, all remaining viewareas are tested again, so that multiple intersections (e.g. VA 3_4_5) are taken into account.

Viewpoints
For a reasonable division of viewareas into viewpoints a 3D grid approach was chosen and the inner area was divided into squares (see Fig. 13).

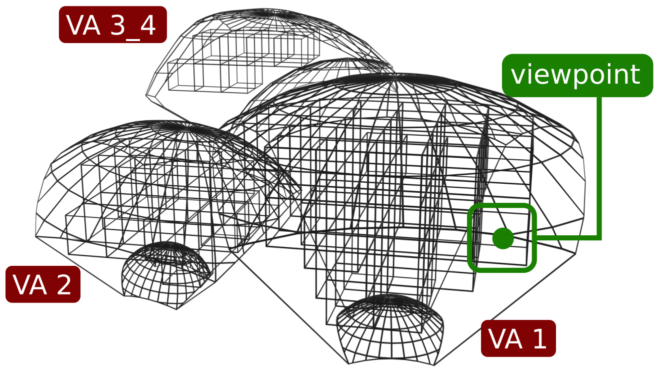


Fig. 13: The three viewareas from image 12 (right) are shown here in a different perspective after the clustering algorithm There are 120 cluster elements in total

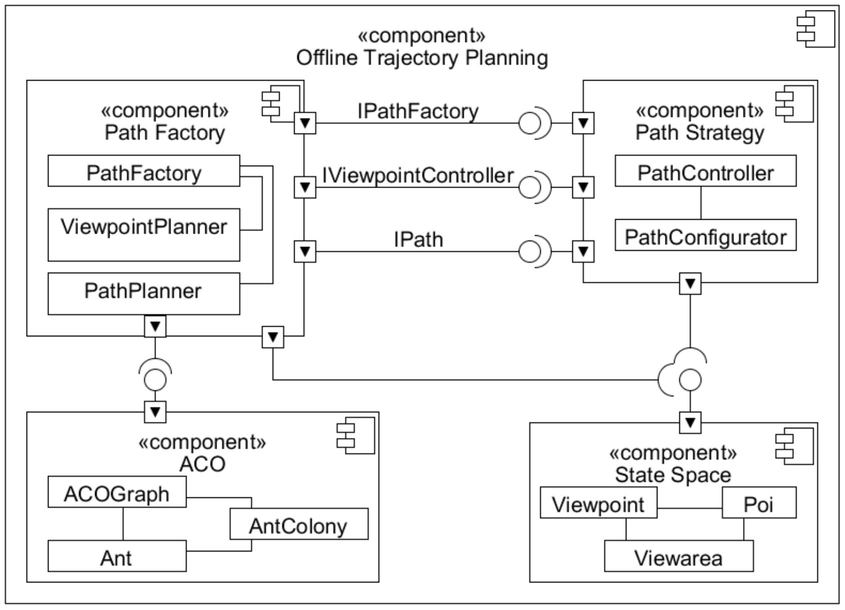


Fig. 14: Component Diagram of the System Architecture of the Offline Planning

The algorithm uses an axis aligned collision hull around the viewareas and divides them into the desired squares of the specified parameters. The collision hull can be enlarged but not reduced. Then there are two alternative procedures. In Fig. 13, the rule is only to use complete inner blocks, so all blocks that are either outside or only partly inside the viewarea will be deleted. Alternatively, only blocks located outside the viewarea are deleted. If the block size is small, the first variant can be used to minimize the number of blocks located at critical edge areas. Accordingly, the center of mass of a block forms the frame of the viewpoint, which is used for trajectory planning.

To calculate the camera orientation(s) at each viewpoint, one key-viewpoint from all possible viewpoints is randomly selected for each viewarea and the orientation is calculated according to the equations 5 to 1 in chapter 3.2. Afterwards the key-viewpoints are handed over to the Trajectory-Planning Part.

4.3 Trajectory-Planning

The Ant Colony system as described in chapter 3.5 was implemented multithreaded. This means that each ant in the colony runs as a separate thread. The Euclidean norm was used as distance measure, because the movement between the viewpoints is considered as linear. The pheromone intensity on each edge is initially set to τ_0:

$$\tau_0 = \frac{1}{\frac{1}{2N} \cdot \sum_{c_{ij}} ||i-j||_2} \tag{12}$$

where N is the number of key-viewpoints. The assignment of the ACS parameters is as follows: $\beta = 5$, $q_0 = 0.5$, $\rho = 0.5$. The amount of ants m is set to $m = \lfloor 0, 3 \cdot N \rfloor$, where N ist the number of key-viewpoints. The implementation of the formula as in Eq. 8, is implemented as a roulette wheel-style selection procedure. This is done by taking the average value of the pheromone and heuristic information of all edges $c_{il} \in N(j^p)$:

$$avg = \frac{\sum_{c_{il} \in N(s^p)} \tau_{il} \cdot \eta_{il}^{\beta}}{N^R} \tag{13}$$

where N^R is the number of viewpoints not yet visited. The edge to the first viewpoint in the list of unvisited viewpoints whose pheromone and heuristic information value of the edge is greater than avg is taken.

5 Evaluation

In the previous chapter we presented the implementation of our concept for planning a path for the inspection of large components using a drone. In this chapter we will evaluate the implementation in two steps. As a first step, a list of viewpoints for the inspection of a sample model with 49 POIs will be calculated. We do this by first generating a viewarea for each POI and then using our optimization strategy to reduce the number of viewpoints by finding intersecting viewareas. The optimized viewareas are used to generate viewpoints and key-viewpoints using the strategy mentioned in section 3. In a second step, we use our ACO implementation to find a near optimal length path visiting all viewpoints. We also compare the length of the two paths with and without the optimized viewareas.

5.1 Viewarea and viewpoint-Planning

For the evaluation of the viewarea and viewpoint planning a fictitious simplified model of an aircraft hull was considered. This model consists of 49 POIs, which are evenly distributed on the inside of the stressed skin. Without viewpoint optimization the 49 POIs result in 49 viewareas with again a trajectory consisting of 49 key-viewpoints. With optimization the number of viewareas and key-viewpoints is reduced to 21. In our use case this is a 57 % reduction of key-viewpoints. Since the drone has to stop at each viewpoint for a certain amount of time to inspect the respective POI, the number of required stops is reduced by just over half, which also reduces the total time required for the inspection. The optimization also significantly minimizes the problem space of the Trajectory-Planning part. A further advantage of viewpoint optimization is that by intersecting the viewareas, they take up less space overall and therefore fewer inner cubes have to be calculated. This means that significantly less time and effort is required for the calculation of viewpoints.

5.2 Trajectory-Planning

For the evaluation of the trajectory the component described in the section before was considered as well. Ten iterations each were performed with the optimized and non-optimized viewpoints and the average was taken as the result. There it could be observed that the optimization not only reduces the number of key-viewpoints, but also shortens the length. The average distance using the optimized ones was 41.3 m and the unoptimized ones 66.2 m. This means that the trajectory length could be reduced by 37.6 % through the viewpoint optimization. The course of the trajectory for optimized and unoptimized viewpoint calculation is shown in Fig. 16.

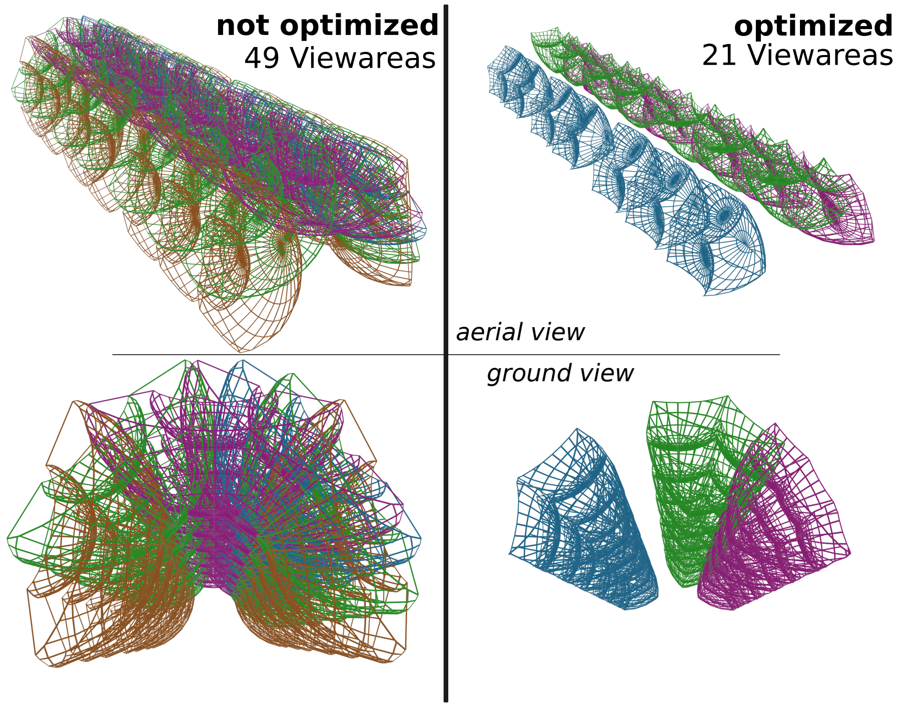
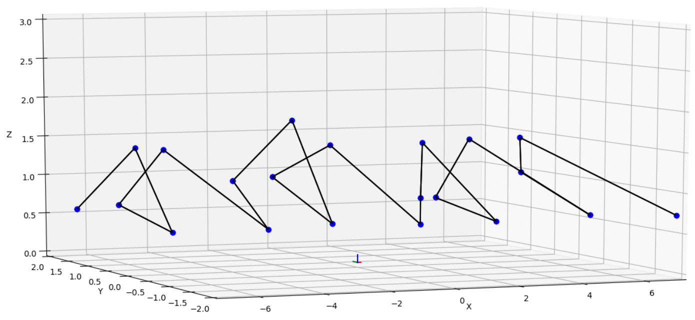


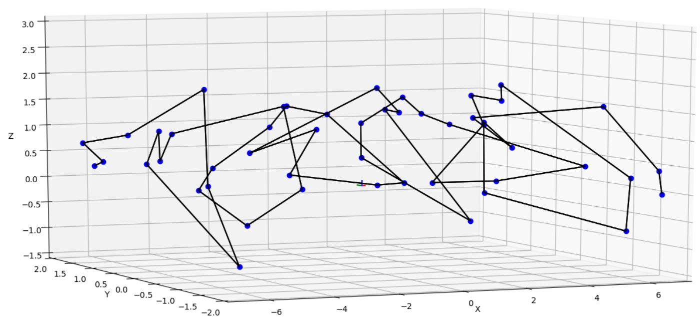
Fig. 15: Result of the unoptimized (left) and optimized (right) viewarea calculation. The optimization reduces the number of viewareas from 49 to 21.

5.3 Ant-Colony-System

The performance of the ACS implementation was evaluated on the TSP data set Att48 [2]. Since this data set consists of two-dimensional coordinates and our implementation is designed for calculation in the three-dimensional, the x and y values were assigned according to the values from the data set and z was set equal to 0. Our Implementation achieved an average distance of 36117.34 after 60 iterations. The best result we achieved on this data set is 34023.35, whose route is shown in Fig. 17.



(a)



(b)

Fig. 16: (a) optimized trajectory consisting of 21 key-viewpoints with a length of 41.3 m and (b) unoptimized trajectory consisting of 49 key-viewpoints with a length of 66.2 m

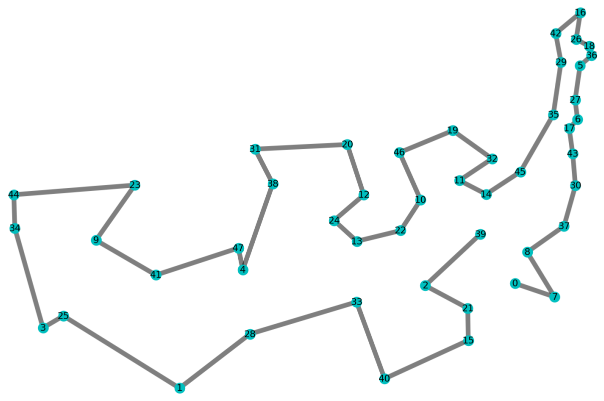


Fig. 17: Course of the route for the distance of 34023.35.

6 Conclusion

In this chapter we demonstrated our approach for inspecting POIs on large structures like an airframe in a production scenario. Instead of using an approach that tries to achieve complete visual coverage of the part to be inspected, we focus on inspecting only predefined POIs of the assembly. Since in a production scenario the changed parts of the assembly between two work steps are well known, it is possible to reduce the overall length of the inspection by focusing on these areas. Our approach generates ViewAreas for each point of interest from which the POI can be seen by the inspecting drone. The number of viewareas is being further reduced by combining intersecting viewareas and a KeyViewpoint is calculated for each ViewArea. Afterwards a Path visiting all KeyViewpoints is planned using the ant colony optimization algorithm. We evaluated our approach using a mockup shell section of an airplane with 49 POIs on the inside. The combination of intersecting ViewAreas resulted in the reduction of the KeyViewpoints on the route from 49 to 21. In future we plan on using the generated path to plan a trajectory for the inspection by a drone. This trajectory doesn't only need to avoid static and dynamic obstacles that could possibly collide with the drone, it also needs to dynamically must also automatically switch to another viewpoint of the viewarea if the view of the POI1 is blocked by an obstacle.

References

1. Diota inspection tool, https://diota.com/en/home, last accessed: 30.10.2020
2. Mp-testdata - the tsplib symmetric traveling salesman problem instances, http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/att48.tsp, last accessed: 25.04.20
3. AG, D.B.: Kompetenzcenter multicopter db, https://www1.deutschebahn.com/dbsicherheit-de/Unsere_Leistungen/Weitere-Leistungen/02_Multicopter-3269322, last accessed: 05.08.2020
4. Alarcon-Herrera, J.L., Chen, X., Zhang, X.: Viewpoint selection for vision systems in industrial inspection. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 4934–4939 (May 2014). https://doi.org/10.1109/ICRA.2014.6907582
5. Bircher, A., Alexis, K., Burri, M., Oettershagen, P., et al.: Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics. In: 2015 IEEE International Conference on Robotics and Automation (ICRA). pp. 6423–6430 (5 2015). https://doi.org/10.1109/ICRA.2015.7140101
6. Danner, T., Kavraki, L.E.: Randomized planning for short inspection paths. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065). vol. 2, pp. 971–976 vol.2 (4 2000). https://doi.org/10.1109/ROBOT.2000.844726
7. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation **1**(1), 53–66 (1997)
8. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **26**(1), 29–41 (1996). https://doi.org/10.1109/3477.484436
9. Englot, B., S. Hover, F.: Sampling-based coverage path planning for inspection of complex structures. ICAPS 2012 - Proceedings of the 22nd International Conference on Automated Planning and Scheduling (06 2014)
10. Galceran, E., Carreras, M.: A survey on coverage path planning for robotics. Robotics and Autonomous Systems **61**, 1258–1276 (12 2013). https://doi.org/10.1016/j.robot.2013.09.004
11. González-Banos, H.: A randomized art-gallery algorithm for sensor placement. Proc. 17th ACM Symp. Comp. Geom. pp. 232–240 (01 2001). https://doi.org/10.1145/378583.378674
12. Jordan, S., Moore, J., Hovet, S., et al.: State-of-the-art technologies for uav inspections. IET Radar, Sonar Navigation **12**(2), 151–164 (2018). https://doi.org/10.1049/iet-rsn.2017.0251
13. Laporte, G.: The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research **59**, 231–247 (06 1992). https://doi.org/10.1016/0377-2217(92)90138-Y
14. LaValle, S.M.: Planning Algorithms. Cambridge University Press (2006). https://doi.org/10.1017/CBO9780511546877
15. Malandrakis, K., Savvaris, A., Domingo, J.A.G., et al.: Inspection of aircraft wing panels using unmanned aerial vehicles. In: 2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace). pp. 56–61 (6 2018). https://doi.org/10.1109/MetroAeroSpace.2018.8453598
16. O'Rourke, J.: Art Gallery Theorems and Algorithms. Oxford University Press, Inc., New York, NY, USA (1987)

17. Sappington, R.N., Acosta, G.A., Hassanalian, M., et al.: Drone stations in airports for runway and airplane inspection using image processing techniques. In: AIAA Aviation 2019 Forum. p. 3316 (2019)
18. Voelcker, H., Requicha, A.: Constructive solid geometry (1977)