UAV Inspection of Large Components: Determination of Alternative Inspection Points and Online Route Optimization

1st Martin Schörner
Institute of Software and Systems Engineering
Augsburg University
Augsburg, Germany
schoerner@isse.de

2nd Constantin Wanninger
ISSE
Augsburg University
Augsburg, Germany
wanninger@isse.de

3rd Raphael Katschinsky
ISSE
Augsburg University
Augsburg, Germany
katschinsky@isse.de

4th Simon Hornung
ISSE
Augsburg University
Augsburg, Germany
hornung@isse.de

5th Christian Eymüller
ISSE
Augsburg University
Augsburg, Germany
eymueller@isse.de

6th Alexander Poeppel
ISSE
Augsburg University
Augsburg, Germany
poeppel@isse.de

7th Wolfgang Reif
ISSE
Augsburg University
Augsburg, Germany
reif@isse.de

***Abstract*—Automation is playing an increasing role in the field of quality assurance. For the visual inspection of larger assemblies such as aircraft fuselages or ship hulls, the use of UAVs is an option. This paper deals with one aspect of the UAV-supported inspection of assemblies in production. Here, newly added components have to be checked for correct assembly. The planning of the shortest possible route from which all components to be inspected can be viewed as well as the estimation of the UAV's position relative to the component have already been presented in previous work. We propose strategies that can be used if an inspection point cannot be reached by the UAV or the component to be inspected cannot be seen by the UAV's camera from the inspection point. For this purpose, we generate alternative inspection points that can be used if errors occur during the inspection from the original inspection point. To achieve this, we present a metric that can be used to select an alternative inspection point that is as suitable as possible. We conclude by demonstrating how this strategy works by evoking different failure cases in a simulated environment.**

***Index Terms*—UAV, inspection, optimization, quality control**

I. Introduction

UAVs, also known as Unmanned Aerial Vehicles, have become a popular tool for a wide range of applications in recent years. They are particularly useful for performing inspections of large structures, such as buildings and infrastructure, because of their ability to safely and efficiently reach areas that may be difficult or dangerous for a human inspector to access. The use of UAVs for inspection purposes has been increasing in various industries and sectors such as construction, energy, transport, oil and gas [1] and many more. Other use cases include the safety inspection of aircraft [2] or ship hulls in dry docks [3]. The use of UAVs in these industries has been able to save time, increase safety, improve the quality of data and reduced cost.

In this paper, we address the problem of fully automating the inspection process using UAVs. Instead of performing a full visual coverage of all parts, we make use of the fact that the parts that have changed in the production process are known and inspect only modified sections of the assembly. We then use these models to guide the UAV to inspect only the necessary parts.

The case study we consider in this paper is the inspection of brackets riveted to the fuselage of an aircraft. We build upon our previous work, in which we have already addressed the challenges of efficiently generating a short route for the UAV to inspect all the necessary parts [4], optimizing the route to prefer points, that view multiple points of interest (POIs) at once [5], and navigating the UAV relative to the structure without the help of external positioning systems [6].

In this paper, we introduce a new approach to inspecting the POIs. We propose a scoring metric that improves when a POI is successfully inspected from a certain point in space and worsens when the inspection fails. We use this metric to search for the most promising inspection routes and determine the best points for the UAV to perform an inspection from (viewpoints, VPs) for each POI. After a successful inspection, the best VPs are determined and a new, optimized route between the VPs is calculated. Overall, the contribution of this paper is a new metric for finding more promising inspection routes over the course of several inspections.

II. Related Work

Autonomous inspection of large components using quadrocopters equipped with a gimbal and view angle dependencies is a rapidly growing field of research. The use of quadrocopters for this task offers several advantages over traditional inspection methods, including improved mobility, flexibility,

and cost-effectiveness. However, the inspection of large components can also present several challenges, particularly when it comes to occluded elements and view angle dependencies. In this chapter, we will review related work in this field and focus on how these challenges have been addressed in previous studies.

One of the key challenges in autonomous inspection of large components is the view area of the mounted camera. To address this challenge, several studies have proposed the use of a gimbal-mounted camera system on the quadrocopter to increase the range of motion and allow for a more comprehensive inspection of the component. For example, in [7], a quadrocopter equipped with a gimbal-mounted camera was used to inspect large wind turbine blades, with the gimbal allowing for a greater range of motion and a more detailed inspection of the blades. Similarly, in [8], a quadrocopter with a gimbal-mounted camera was used to inspect large aircraft wings, with the gimbal allowing for a more comprehensive inspection of the wings, including the underside.

Another key challenge in autonomous inspection of large components is the variation in view angle dependencies considered in the route planning. This can be caused by the geometry of the component or by static obstacles in the environment. To address this challenge, several studies have proposed the use of dynamic trajectory planning algorithms to adjust the quadrocopter's flight path and camera angles. For example, in [9], an algorithm was proposed that uses a combination of visual odometry and a 3D map of the component to plan a dynamic flight path for the quadrocopter, allowing it to adjust its camera angles to inspect different parts of the component. Similarly, in [10], an algorithm was proposed that uses a 2D map of the environment to plan a dynamic flight path for the quadrocopter, allowing it to avoid obstacles and maintain a clear view of the component.

Another important consideration for autonomous inspection of large components is the ability to operate in indoor scenarios with dynamic obstacles, such as humans. In [11], a quadrocopter equipped with a gimbal-mounted camera was used to perform autonomous inspection in indoor environments with obstacles, with the algorithm using a combination of visual odometry and collision avoidance algorithms to plan a dynamic flight path and avoid collisions. Furthermore, in [12], a quadrocopter equipped with a gimbal-mounted camera and a LIDAR sensor were used to perform autonomous inspection in indoor environments, with the sensor providing 3D mapping of the environment and helping to avoid collisions.

In conclusion, autonomous inspection of large components using quadrocopters equipped with a gimbal and view angle dependencies is a complex task that requires a combination of advanced algorithms and hardware. Previous studies have addressed the challenges of occluded elements and view angle dependencies through the use of a gimbal-mounted camera system and dynamic trajectory planning algorithms, while also considering indoor scenarios with dynamic obstacles. Our approach is not only dynamic planning, but also optimization of the route over runtime in the event of unplannable occlusions in an indoor scenario (i.e. without external positioning systems).

III. Concept

This section gives an overview of the underlying concept of the work. First, the terminology of points of interest (POIs), viewpoints (VPs) and viewareas (VAs) is described. We then discuss how viewpoints are generated and how the optimal viewpoint can be chosen to inspect a POI. In particular, we discuss how experience from previous inspections can be used to optimize future inspections.

A. POIs, Viewpoints, Viewareas

The three key concepts of our architecture are points of interest, viewpoints and viewareas. These have already been introduced in our previous work [4], but are briefly defined again here for the reader's better understanding. POIs are points to be inspected at an assembly (see: fig. 1). These can be, for example, components added in the previous production step, whose correct assembly is to be checked. A geometric volume, the VA, can be calculated using the aperture angle of the inspection UAV's camera and by defining a minimum and maximum distance from which the component can be optimally seen. If the UAV is in the VA, it can theroetically sees the POI to be inspected from a distance that allows for inspection. Points from which a POI can be inspected are called VPs. They can be created by selecting arbitrary points within the VA. Each VP also contains an orientation that the drone and its camera must adopt in order to look at the POI. Since the information about the orientation is not required for the concepts in the rest of the paper, VPs are only considered as points in 3D space. In order to reduce the number of VPs, several overlapping VAs can be combined into one optimized VA by intersecting the two volumes. All POIs of the original VAs can then be seen from the new VA. This means that several POIs can also belong to one VA. For more information on this process, please refer to our previous work [5].

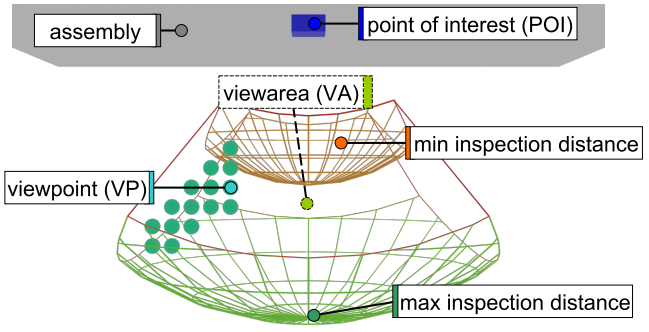

Fig. 1. A viewarea, viewpoints and point of interest on an assembly that needs to be inspected

B. Alternative VP generation and Key Viewpoints

Due to the definition of VAs as volumes, there is theoretically an infinite number of VPs to choose from. This causes

the problem that a finite number of VPs that are best suited an for the inspection and dissimilar from each other so that if inspection from one viewpoint fails, chances are high that another one will lead to a different result. Even though any VP within a VA can theoretically be used for inspection, in practice it may happen that one or more VPs are not accessible, e.g. due to an obstacle blocking the path to the VP, or the view from VP to the POI being obscured by unforeseen obstacles. This causes the problem of selecting a suitable alternative VP.

Since in these cases approaching a VP that is close to the previous one will probably not solve the problem (because nearby points are most likely affected by the same obstacle as well), we reduce the number of possible VPs by selecting a 3D grid of points at fixed distances from each other from the points of the viewarea (see: fig. 1). This allows us to guarantee a certain minimum distance between the points, and also limits the choice of VPs to a finite set. The exact operation of the procedure has already been described in past works [5]. For the creation of an inspection route of several POIs, one VP is selected for each VA as a key viewpoint, which is approached during the inspection. All other VPs of the VA serve as alternative VPs if an inspection via the key VP is not possible. Finding the shortest route between the key VPs is considered a Traveling Salesman problem. We have already presented our solution for this in [4].

C. Selection of optimal Viewpoints

As described in the previous section, the order of the POIs to be inspected is defined in the precalculated route and one VP per VA is suggested. The mission planning process handles the VAs in sequence and begins by selecting the VP with the highest score from the current VA. A the VP's score is continually updated during the inspection process (see fig. 2). An attempt is then made per VA to fly to the selected VP from which all POIs in the VA can be inspected. If not all POIs can be inspected from one VP, another VP of this VA is selected. The selection of the next VP is based on a combination of factors, including its score (VPs with higher scores are preferred) and the distance to the VP where the UAV is currently located. The latter factor is included in order to get a VP that is far away from obstacles blocking the view from the previous VP as possible. Before selecting the next VP, the score of the previous VP is reduced. This procedure is repeated until all POIs of the VA have been inspected. If all POIs of a VA can be inspected from a VP, its score is increased accordingly and the inspection of the next VA is continued.

1) Scoring System: To determine the quality of a VP, each VP receives a score that is initialised with 0. After each approach attempt of the UAV to the VP, the score is updated.

If a component is successfully detected from a VP, the score is increased and the VP is visited with a higher probability the next time.

If a POI cannot be approached by the UAV, for example because it is blocked by an obstacle, its score is reduced and another VP with the highest possible score and at a greater distance from the current VP is approached.

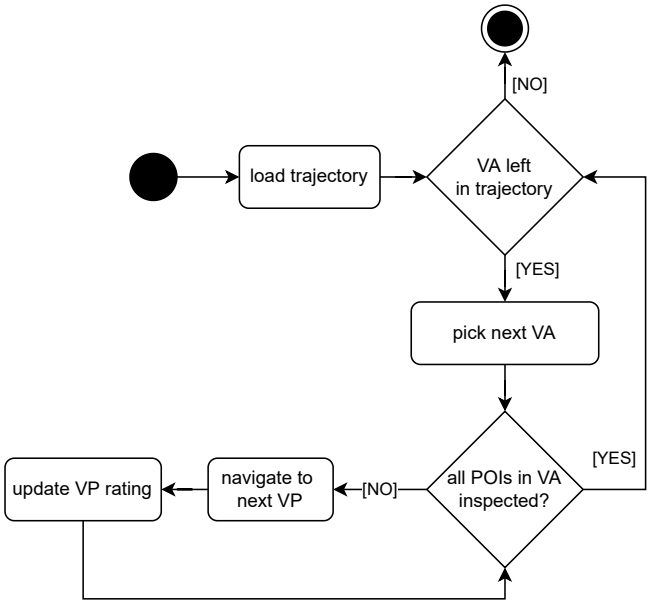

Fig. 2. Overview of the processing of VAs during inspection

Figure 3 describes how to decide whether a POI can be inspected from a VP. Its logic is located in the node "Navigate to next VP" in Figure 2. First, using the Euclidian Distance from the current UAV position to the VP and the approximate estimated speed of the UAV, a time is calculated in which the target should be reached. If the UAV exceeds this time and an additional buffer, the target probably cannot be reached at all and another VP is approached. In case a VP cannot be reached and the first timout is triggered, its score is reduced by a parameter A.

$$score := score - A \tag{1}$$

If a VP is reached, a second timer is started, waiting for all POIs to be detected by the inspection software. If this does not happen within the set time limit, an attempt is made to fly to another VP. In case of the timeout being triggered, the VP's score is also reduced by a parameter B.

$$score := score - B \tag{2}$$

In addition, before the VP is approached, we determine the Euclidean distance d_e between the VP and the UAV. During the flight, the distance actually covered by the UAV d_r is measured. In order to penalise unnecessarily long distances with detours around e.g. obstacles, the additional distance flown compared to the Euclidean distance is considered and included in the score. For this purpose, the VP's score is updated as follows when it is successfully reached:

$$trajectory\_efficiency := \frac{d_e \cdot 1.1}{d_r} \tag{3}$$

$$score := score + C \cdot trajectory\_efficiency - 1 \tag{4}$$

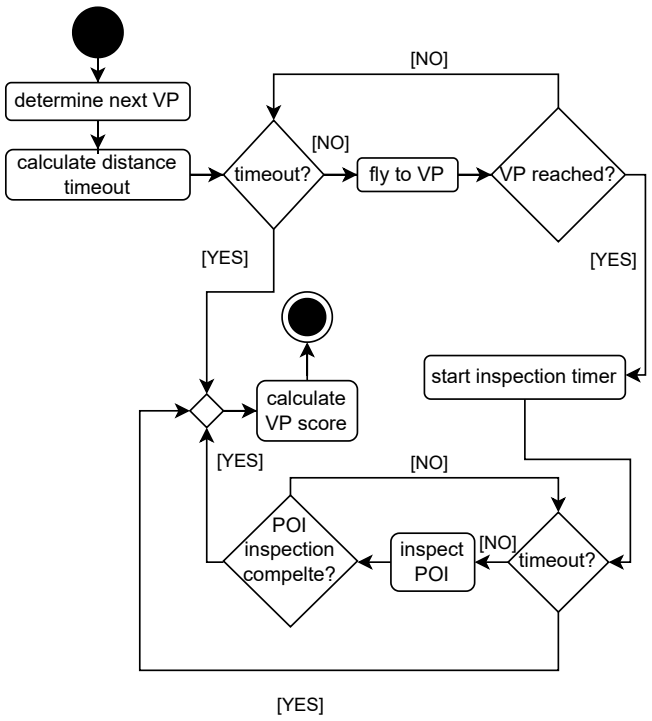

Fig. 3. Mechanics of the failure detection and scoring system

Where C represents a parameter weighing this metric against the other two. The Euclidean Distance is multiplied by a factor of 1.1 because the UAV can never travel the perfect path between two points due to control deviations. This factor avoids penalising this behaviour.

Upon successful inspection of all POIs from a VP, it's score is increased as follows:

$$score := score + D \tag{5}$$

Where D is another constant parameter. The factors A, B, C and D allow the influence of the various error cases and a successful inspection on the development of the score of a VP to be weighted.

Finally, the value of the score is limited to both a maximum and a minimum value. This is done to prevent a point from receiving an extremely high score it was approached successfully many times. This would mean that it would take an extremely long time to switch over to an alternative VP if, for example, a VP with many successful inspections could not be reached anymore due to an obstacle.

D. Optimization of inspeciton route with data from last inspection

After each inspection, the viewpoint with the highest score in each viewarea is selected as the new key viewpoint for the next inspection. Afterwards, our implementation of the AntNet algorithm [4] calculates the optimal visit order for the new key viewpoints again. This results in a new, optimized route with known good key VPs that can be used in the next inspection of this type of assembly.

IV. Implementation

Implementation was done in the already existing system described in [6]. The system uses the ROS framework [13]. The evaluation of the camera images for the actual inspection is carried out by the software DIOTA [14]. For the simulation, the software RotorS [15] is used.

A. System Components

The system consists of three main components: Insepction, ROS and UAV. The Inspection component is responsible for evaluating the video stream coming from the UAV and estimate the pose of it. Both is done by the Inspector subcomponent. It receives the video stream from the UAV component via RTMP, evaluates it, estimates the UAV's relative pose based on the video stream and sends the pose estimates to the ROS component via the rosserial component. The ROS component takes care of monitoring an inspection, i.e. planning VPs and navigating the UAV at runtime. The QuAD Planning component is responsible for calculating routes and providing route-specific information to the QuAD Control component. The latter uses this information to perform an inspection. Further details about the route execution by the QuAD Control component is explained in more detail in section IV-B. The UAV component sends odometry data via the Mavlink Adapter Component to the ROS System and receives control commands throught it. Among other things, it sends the velocities of the UAV to be fused with the camera tracking of the Inspection component to estimate the UAV's pose and obstacle detection data to estimate its state and avoid obstacles during navigation.

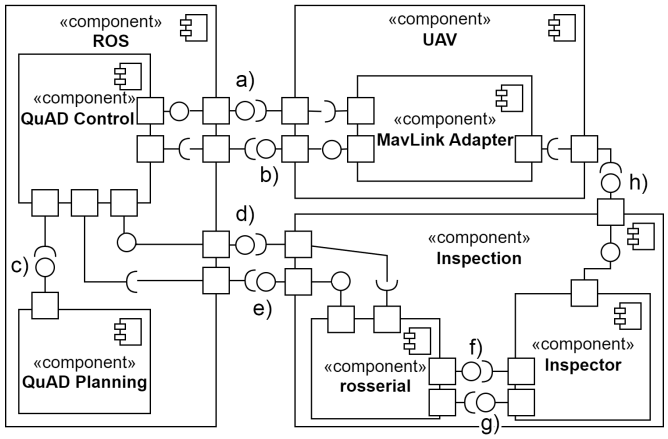

Fig. 4. Components of the system and its interfaces: The interfaces a) and b) are responsible for sending and receiving MavLink messages. Interface c) is used by the QuAD Control component to retrieve route information. Interface e) sends the filtered UAV pose of the ROS component to the Inspector component via interface g). Interface f) forwards the UAV's pose estimations of the Inspector component to the ROS component via interface d). The Inspector component accesses the video stream of the UAV via interface h).

B. ROS Architecture

As shown in Figure 5, the heart of the system is the /mission_control node, which is responsible for controlling and monitoring an inspection. It supervises all control commands for the UAV, except for the target speed commands and camera gimbal commands, and transmits them to the /copter_manager node for further processing. It monitors the inspection by controlling the sequence and safety of the inspection. To determine the sequence, it requests trajectory-specific data from the trajectory_control node, selects the viewpoints to be targeted, and requests alternative viewpoints from the /trajectory_control node if necessary. To monitor the inspection's safety it instructs watchdogs (not shown in the diagram for the sake of clarity) to monitor obstacles, camera tracking, connection, etc. to intervene in the inspection if necessary and protect the component and UAV from damage. In addition to calculating (alternative) viewpoints, the /trajectory_control node also provides a service for updating the viewpoints rating to the /mission_control node. The /copter_manager node serves as a communication interface between the UAV and ROS. Among other data, it receives data relevant for navigation, such as obstacles in sight and velocites from the UAV via the topic /mavlink/from. Obstacles are forwarded to the /quad/obstacle topic and velocities to the /quad/current_vel topic, where they can be further used for navigation. It receives the UAV's speed and gimbal commands directly from the topics cmd_vel and /quad/oreintation_gimbal. Since both commands are continuously calculated and published by the system it makes more sense to get them directly from the topics instead of calling them through the /mission_control node by e.g. a service or action. The speed commands for the uav and the gimbal commands for the camera are then forwarded by the /copter_manager to the topic /mavlink/to in the form of MavLink commands. All other control commands are called by the /copter_manager as ROS action calls. The /camera_tracking is an external node that processes the camera data of the UAV and continuously publishes pose estimates of the UAV in the Topic /tf, as well as information about the component and POIs in /quad/comp_info.

As shown in Figure 6, for navigation, the /mission_control node transmits the next viewpoint to the /orientation_to_comp_publisher and /nav_cmd_publisher nodes. Since each VP requires an individual orientation of both the UAV and its camera gimbal to correctly capture the POI, the /orientation_to_comp_publisher node continuously obtains the UAV's current pose from the /tf topic and calculates the orientation for the UAV's camera gimbal. The UAV's orientation can then be obtained by the node /nav_cmd_publisher via the topic /quad/orientation to calculate the target velocities to reach a VP's pose and publish them to the topic /cmd_vel. The node /nav_cmd_publisher also includes an obstacle avoidance system using the potential field method [16] wich we described in detail in previous work [6]. The orientation for the camera gimbal is published by the node /orientation_to_comp_publisher on the /quad/orientation_gimabl

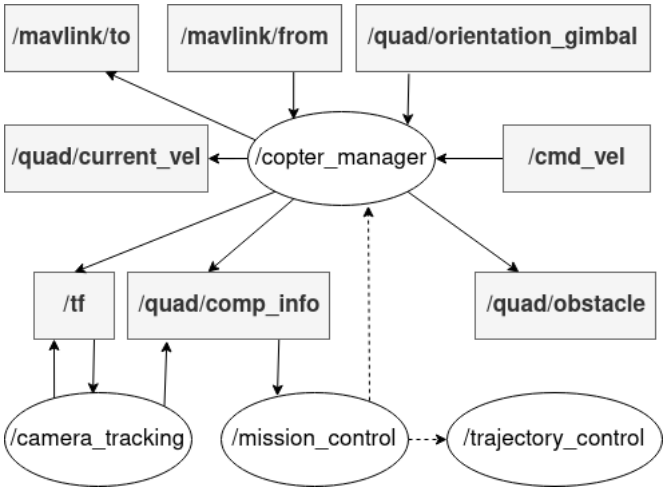

Fig. 5. Node Architecture for monitoring an inspection.

topic, where it can be used, as shown in Fig. 5 and mentioned above, by the /copter_manager node. In order to take obstacles into account during navigation, the node /nav_cmd_publisher additionally obtains data about them via the topic /quad/obstacle. The UAV's current pose is calculated by the node /state_estimation_publisher and continuously published to the topic /tf. For the estimation of the UAV's current pose the node /state_estimation_publisher gets the pose estimates from the camera tracking via the topic /tf and the UAV's current speed via the topic /quad/current_vel.

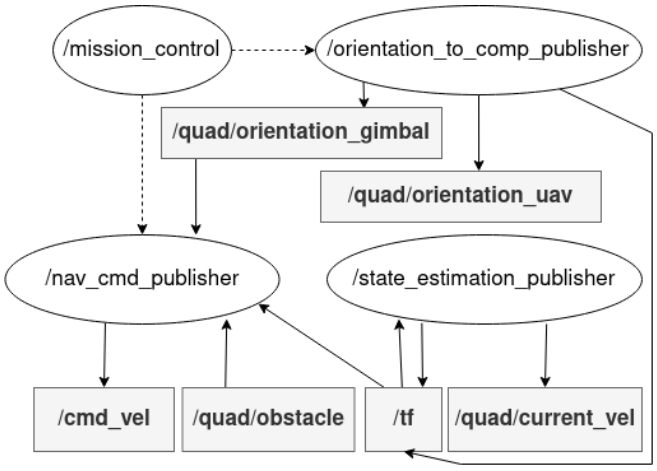

Fig. 6. Node Architecture for state estimation and calculation of navigation commands.

V. Proof of Concept

In this section, we present the results of our proof of concept study, which aimed to demonstrate the feasibility of our proposed method for fully automating the inspection process using UAVs. The study was conducted in a virtual test environment, which included a component to be inspected and obstacles that the UAV needed to navigate around.

A. Experimental setup

The experiment was conducted in simulation, in which a component to be inspected containing three POIs were present (see fig. 7). In some of the scenarios, additional obstacles were added in the flightpath of the UAV. The goal of the experiment was to demonstrate the ability of the UAV to inspect the various Points of Interest (POIs) on the component. To accomplish this, a route was planned in advance using the method described in our previous work. The simulated inspection software was used to recognise the components and detect any inspection problems at arbitrary viewpoints (VPs).

Three different scenarios were tested in the proof of concept. The first aimed to evaluate the UAV's reaction to an obstraction of the camera's view of the POI, while the second aimed to evaluate reactions to the inability of the UAV to reach a VP that is obstructed by an obstacle. Finally, both problems were tested together in a third scenario.

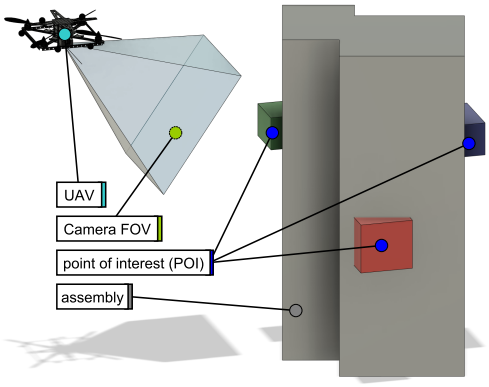

Fig. 7. Setup for the proof of concept experiments. An UAV equipped with an inspection camera tries to inspect three different parts mounted on an assembly

B. Scenario 1:

In this experiment, we aimed to demonstrate the ability of the UAV to react to obstacles between the UAV's camera and a POI causing the inspection to timeout. We first assumed that there were no obstacles on the route of the UAV. We then used simulated inspection software to cause inspection problems at an arbitrary viewpoint. The goal of this experiment was to show that the UAV could register that the point of interest (POI) could not be seen from the current VP, fly to another one and, after a few attempts, replace the VP from which the POI could not be seen.

The results of the experiment showed that the UAV initially flew to a VP centrally located in the VA and attempted to carry out an inspection of the POI from there (see fig. 8). After 10 seconds, the timeout for the inspection of the component occurred, and the UAV selected a distant VP of the same VA, taking into account the score. This VP was flown to, and the inspection of the POI was successfully completed. Finally, the scores of the approached VPs were updated. When the inspection process was repeated, it was observed that the UAV directly flew to the newly determined, unshaded VP (see fig. 9), which resulted in a shorter route execution, from 34.3 to 31.7 m and from 176 to 110 s.

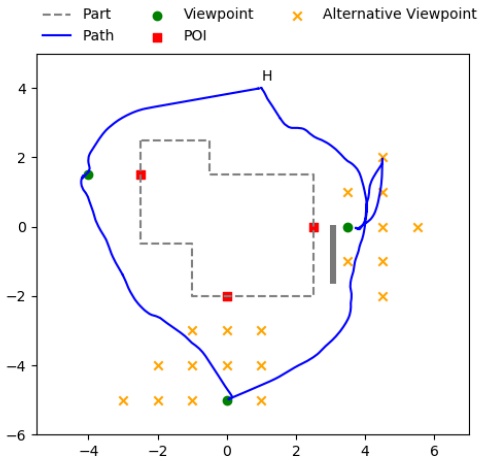

Fig. 8. Trajectory of the first inspection flight of the first scenario. The right POI can not be inspected from the initial key VP. Therefore anoter one is chosen. "H" marks the starting and landing point and the UAV is flying in a clockwise direction.

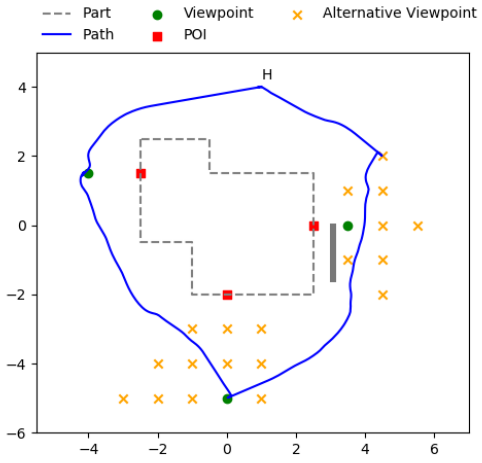

Fig. 9. Trajectory of the second inspection flight of the first scenario. The initial key VP for the right POI is not visited anymore. Instead the UAV navigates directly to the previously successfull VP. "H" marks the starting and landing point and the UAV is flying in a clockwise direction.

C. Scenario 2:

In this experiment, we aimed to demonstrate the UAV's ability to avoid VPs that are unreachable. We added obstacles on the route that prevented one of the POIs from being reached.

The results of the experiment showed that the UAV registered that a VP could not be reached, flew to another VP and

replaced the VP from which the POI could not be seen on the next attempt (see fig. 10).

When the system realized that it could not reach a VP, the attempt to fly to the VP was aborted and, according to our metric, another VP of the same VA was selected. This VP was flown to and enabled the inspection of the POI. Finally, the scores of the approached VPs were updated again. Here too, we saw that the newly determined VP that was not obstructed by an obstacle was approached directly during the re-inspection (see fig. 11). The route execution was shortened from 33.6 to 32.6 m and from 217 to 158 s.

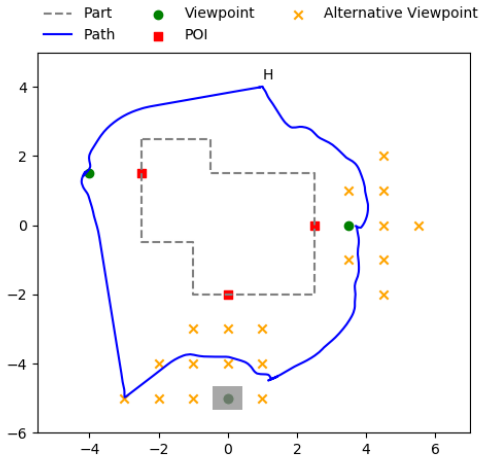

Fig. 10. Trajectory of the first inspection flight of the second scenario. The UAV fails to reach the key VP for the bottom POI and moves to an alternative VP after a timeout occurs. "H" marks the starting and landing point and the UAV is flying in a clockwise direction.

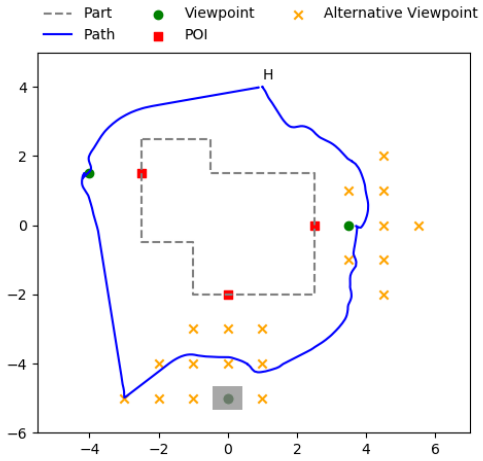

Fig. 11. Trajectory of the second inspection flight of the second scenario. The higher score of the Alternative VP of the bottom POI visited in the previous flight causes the UAV to navigate directly to it instead of the old key VP. "H" marks the starting and landing point and the UAV is flying in a clockwise direction.

D. Scenario 3

Scenario 3 combines the problems of the first two Experiments into one inspection. For the POI on the right, an obstacle that shadows the camera's field of view prevents an inspection of the part from the original key VP (see fig. 12). Therefore, its score is first reduced before a new key viewpoint is searched. Since all viewpoints are still initialized with 0, the one with the largest distance to the old key viewpoint is selected. The inspection can be successfully performed from this viewpoint, which increases its score and it will be approached immediately on the next pass without visiting the original key viewpoint first.

For the second POI, an obstacle prevents the originally defined key viewpoint from being reached. Again, after the problem is detected, the score of the original key viewpoint is reduced. Then the viewpoint with the greatest distance to the old VP is also approached, since the scores of the VPs are all zero here as well. After the reachability of the new viewpoint is not affected by the obstacle, the inspection from this point is successful and the score of the VP is increased. Again, it can be seen that due to the updated scores, the UAV automatically selects the new viewpoint as the key viewpoint in the second pass and does not approach the original one (see fig. 13).

In this experiment, the distance traveled by the UAV was reduced from 36.8 to 33.2 m and the execution time was reduced from 267 to 159 s.

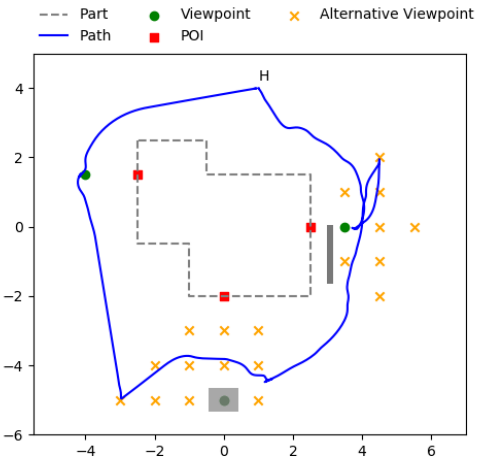

Fig. 12. Trajectory of the first inspection flight of the third scenario. The right POI can't be inspected from it's initial keypoint, therefore the UAV choses different one. Like in scenario 2, the key VP of the bottom POI cannot be reached due to an obstacle. As a result, the system choses one further left. "H" marks the starting and landing point and the UAV is flying in a clockwise direction.

In summary, we were able to show with our series of experiments that our concept reacts to obstacles and obstructed POIs during simple inspections and, by adjusting the scores, calculates an optimized route in the next run which causes the UAV to no longer fly to previously ineffective VPs.

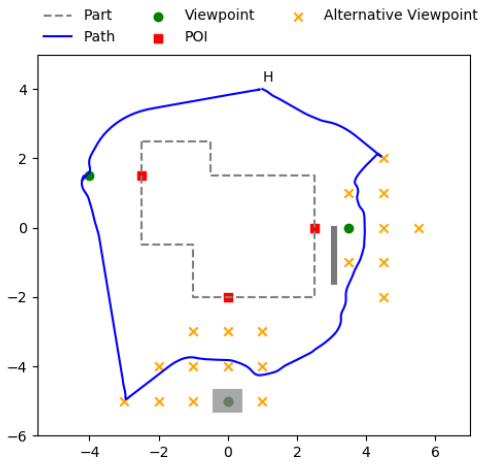

Fig. 13. Trajectory of the second inspection flight of the third scenario. Due to the updated scores, the UAV directly navigates to the previously successfull VPs and skips the original key VPs. "H" marks the starting and landing point and the UAV is flying in a clockwise direction.

VI. Conclusion

This paper addressed one aspect of inspecting a large structure such as a ship hull or aircraft fuselage using a UAV. In particular, the concept of selecting alternative inspection points that can be used when a VP either cannot be reached or the inspection from the original VP fails was discussed. For this purpose, an algorithm was presented that uses a scoring system to reward successful inspection points and penalize points that either cannot be reached, where the inspection fails, or those where the UAV has to fly a large detour to approach with a poor score. In addition, a distance as far as possible from the original VP is included in the selection criteria, in order to get as far away as possible from obstacles or obstructions of the view of the POI that exist at the original viewpoint. This scoring system allows to avoid "bad" VPs on the next inspection and to fly a more efficient and faster route on the next inspection flight.

This concept was integrated into our existing system and its implementation in ROS was described. Subsequently, a proof of concept was performed using three scenarios. In the first one a POI cannot be inspected from a VP, in the second one the VP cannot be reached due to an obstacle. Finally, both error cases were considered together in one scenario.

In the proof of concept it can be seen that the UAV uses the scoring system to mark successful and unsuccessful VPs. In the second run, the previously unsuccessful VPs are no longer approached, but the VPs that were successful in the last run are used directly, and both the execution time and the distance flown are shortened compared to the previous run. The influence of an obstacle blocking the direct path from one VP to the next and leading to flying a detour was not evaluated. Again, the VP should be given a worse score because the actual distance flown is longer than the distance as the crow flies. However, this is the subject of future work.

ACKNOWLEDGEMENTS

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

References

[1] D. B. AG, "Kompetenzcenter multicopter db," last accessed: 20.01.2023. [Online]. Available: https://www.dbsicherheit.de/dbsicherheit-de/Unsere_Leistungen/Weitere-Leistungen/02_Multicopter-7749316

[2] R. N. Sappington, G. A. Acosta, M. Hassanalian *et al.*, "Drone stations in airports for runway and airplane inspection using image processing techniques," in *AIAA Aviation 2019 Forum*, 2019, p. 3316.

[3] B. Englot and F. S. Hover, "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.

[4] C. Wanninger, R. Katschinsky, A. Hoffmann, M. Schörner, and W. Reif, "Towards fully automated inspection of large components with uavs: Offline path planning," 2020.

[5] M. Schörner, R. Katschinksy, C. Wanninger, A. Hoffmann, and W. Reif, "Towards fully automated inspection of large components with uavs: offline path planning and view angle dependent optimization strategies," in *Informatics in Control, Automation and Robotics: 17th International Conference, ICINCO 2020, Lieusaint - Paris, France, July 7–9, 2020, Revised Selected Papers*, O. Gusikhin, K. Madani, and J. Zaytoon, Eds., 2022.

[6] M. Schörner, M. Bettendorf, C. Wanninger, A. Hoffmann, and W. Reif, "Uav inspection of large components: indoor navigation relative to structures," in *Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics - ICINCO, July 6-8, 2021*, O. Gusikhin, H. Nijmeijer, and K. Madani, Eds., 2021.

[7] A. P. Castro, J. R. R. Alves, and P. J. G. Costa, "Autonomous inspection of wind turbine blades using a quadrocopter equipped with a gimbal-mounted camera," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5807–5815, 2018.

[8] Z. Liu, C. Hu, and X. Zhang, "Autonomous inspection of large aircraft wings using a quadrocopter equipped with a gimbal-mounted camera," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5816–5824, 2018.

[9] Y. Wang, X. Chen, and W. Zhang, "Dynamic trajectory planning for autonomous inspection of large components using a quadrocopter equipped with a camera," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 1–13, 2018.

[10] S. Sun, J. Liu, and L. Zhang, "Obstacle avoidance and dynamic trajectory planning for autonomous inspection of large components using a quadrocopter," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 7, pp. 5683–5691, 2019.

[11] J. Kim and B. Park, "Autonomous inspection of indoor environments with obstacles using a quadrocopter equipped with a gimbal-mounted camera," *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 14–23, 2018.

[12] J. Zhang, Y. Wang, and X. Hu, "Autonomous inspection of indoor environments using a quadrocopter equipped with a gimbal-mounted camera and a lidar sensor," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 7, pp. 5692–5700, 2019.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[14] "Diota inspection tool," last accessed: 20.01.2023. [Online]. Available: https://diota.com/en/home

[15] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," in *Robot operating system (ROS)*. Springer, 2016, pp. 595–625.

[16] Y. Koren, J. Borenstein *et al.*, "Potential field methods and their inherent limitations for mobile robot navigation." in *ICRA*, vol. 2, 1991, pp. 1398–1404.