Autonomous Visual Inspection of Industrial Plants Using Unmanned Aerial Vehicles

Vincenzo Scognamiglio, Riccardo Caccavale, Pasquale Merone, Alessandro de Crescenzo, Fabio Ruggiero, Vincenzo Lippiello

***Abstract*— The development of autonomous systems has spurred numerous innovative inspection strategies. Some operations, such as monitoring the condition of industrial structures, typically entail significant deployment of human resources and pose risks to human safety. In this context, this paper presents a visual inspection framework that leverages unmanned aerial vehicles to explore designated facilities, identifying structural damages such as cracks or fissures for inspection. The proposed approach integrates autonomous navigation and high-level decision-making capabilities to effectively explore predefined points of interest within partially known environments and to select and inspect candidate spots for further analysis. The framework is validated through both simulated and real-world experiments conducted in GPS-denied environments, utilizing only onboard UAV capabilities.**

I. INTRODUCTION

The advent of new technologies in autonomous systems presents an opportunity to deploy them in real-world scenarios to assist in tasks deemed hazardous and repetitive for human operators. This is particularly evident in the monitoring of industrial plants, which require ongoing maintenance to ensure safe working environments for personnel. Often, such inspections pose challenges for human operators, especially in cases where the plant's infrastructure is complex, elevated, and difficult to access (see Fig. 1). These operations serve various purposes depending on the specific plant being inspected. For instance, there may be a need to detect and localize corrosion in steel structures, as failure to do so promptly could compromise plant performance and pose safety risks. Similarly, inspections of concrete facilities aim to identify areas of damage or cracks that, if left unaddressed, could create hazardous conditions.

Critical inspection points are often situated in locations that are not easily visible, at great heights, or otherwise inaccessible to human operators, who require specialized training and equipment for such tasks, incurring significant costs. In this context, the utilization of autonomous aerial platforms equipped with vision sensors emerges as a cost-effective and reliable solution for inspecting and monitoring inaccessible sites. Aerial platforms' ability to maneuver in six degrees of freedom (DOFs) enables them to reach locations with limited access, while low-cost cameras allow for visual inspections to identify cracks, corrosion, or other defects.

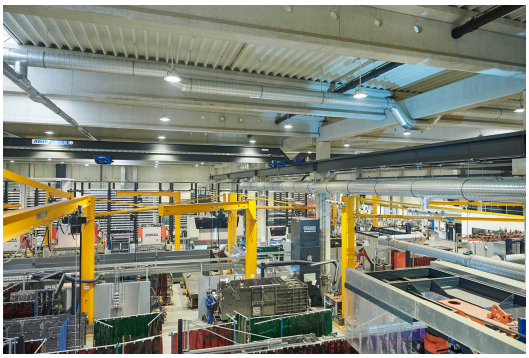


Fig. 1: Automotive industrial plant: environment in which the proposed framework is meant to work

The primary contribution of this work lies in the development of a fully autonomous method for deploying unmanned aerial vehicles (UAVs) in visual inspection tasks. The UAV autonomously detects inspection spots, simulated with ArUco markers [1], plans its movements to approach these spots, and inspects them using an onboard camera. To validate the approach, simulated case studies in an industrial plant-like scenario based on Gazebo are conducted, along with preliminary real-world experiments in an indoor GPS-denied environment to demonstrate its effectiveness.

The remainder of the paper is organized as follows: Section II provides an overview of existing methods for visual inspection of industrial plants. Section III presents the proposed approach, focusing on the customized navigation software implemented and the autonomous decision-making algorithm developed for the monitoring operation. Section IV describes the simulated and real-world tests, along with their results. Finally, Section V concludes by summarizing the case study results and discussing potential future developments.

II. RELATED WORKS

The increasing prevalence of autonomous aerial platforms is attributed to their ability to replace human operators in hazardous and repetitive tasks, such as the inspection of industrial structures, which often incur high costs and require extensive human involvement. A significant body of literature focuses on utilizing drones for monitoring industrial plants. In [2], the authors categorize non-destructive inspection approaches based on the type of industrial structure. For

The research leading to these results has been supported by the AI-DROW project, in the frame of the PRIN 2022 research program, grant number 2022BYSBYX, funded by the European Union Next-Generation EU, and the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie (grant agreement n. 953454). The authors are solely responsible for its content.

[1]All the authors are with PRISMA Lab and CREATE Consortium, Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples, Italy. vincenzo.scognamiglio2@unina.it

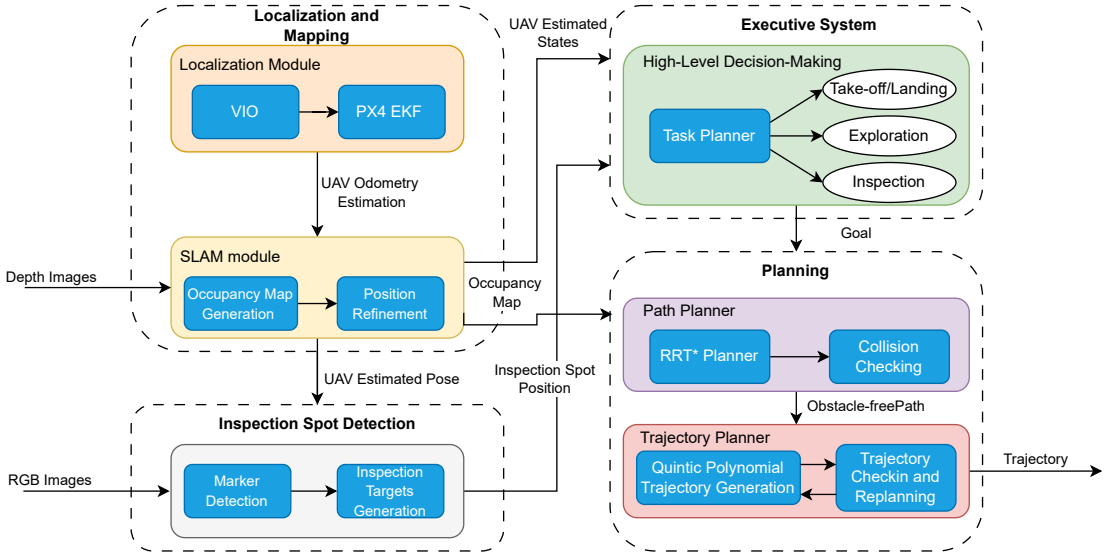


Fig. 2: Autonomous inspection scheme. The framework is composed by four modules, one related to the perception of the environment and localization, the second allows the aerial robot to find inspection spots and to generate inspection targets. The executive system decides which task the drone has to carry out and sends target positions to the planner that generates the executed trajectories.

instance, in the inspection of photovoltaic power systems, thermal cameras are employed to detect failures [3]. For power-line inspection, deep learning methods are utilized with a gimbal to detect defects [4], and in [5], a multirotor capable of semi-automatically landing on power lines is designed. Drones have also been utilized in mining industries, such as in [6], where they are used for semi-automated thermographic inspection of belt conveyors. Wind turbines, another type of plant requiring frequent inspection, have been studied extensively, with approaches like using convolutional neural networks for optical inspection of rotor blades [7]. Concrete structures within industrial plants are another area where drones are frequently deployed for inspection. In [8], a method based on YOLO [9] is developed to detect and estimate the location of cracks. Drones are also useful for inspecting steel structures, including pipe racks typical of the oil and gas industry [10], and in [11], UAVs are utilized to inspect pipes and detect possible corrosion signs. In the majority of the works discussed, drone operations are conducted in a semi-autonomous manner, wherein one or more operators teleoperate the platform to identify inspection spots and visually inspect them using pre-planned trajectories. This paper introduces a framework for deploying UAVs to perform visual inspection tasks enhancing the full autonomy capabilities of the platform concerning specific detection methods. Therefore, the proposed framework limits the operators' tasks to commanding mission initiation and termination, leaving almost complete decision-making autonomy to the aerial platform. In scenarios where the environment is semi-unknown, the drone explores predefined points of interest without the help of human operators. Upon detecting an inspection spot, recognized by an ArUco marker, the system generates five inspection waypoints and autonomously decides whether to explore them or continue with a previously scheduled task.

III. PROPOSED SOLUTION

A. Overview

The system comprises four main modules (see Fig. 2).

1) *Localization and Mapping*. Implemented onboard the UAV, this module perceives the surrounding environment and provides an occupancy map along with the refined pose of the robot.
2) *Inspection Spot Detection*. Utilizing red-green-blue (RGB) images and the UAV's pose, this module identifies fiducial markers and generates targets for the inspection task.
3) *Executive System*. Controls the high-level behavior of the platform by generating target positions, which are then fed into the planning module.
4) *Planning*. Receives target positions from the executive system and generates trajectories for the execution.

The whole system is designed to run onboard the UAV computer which has the duty of communicating with a ground control station where an operator can start and finish the mission. The subsequent sections provide a detailed exploration of each module's functionality.

B. Localization and Mapping

In the proposed approach, the aerial platform relies solely on onboard capabilities for stable autonomous flights over extended periods. To achieve this, a simultaneous localization and mapping (SLAM) algorithm is essential [12]. This algorithm enables the drone to construct a map of its surrounding environment while simultaneously estimating its position in a world fixed frame.

Odometry, crucial for navigation, is computed using methods involving stereo cameras and inertial measurement units

(IMUs), such as visual-inertial odometry (VIO) algorithms. In the proposed solution, odometry estimation from the VIO submodule feeds into the extended Kalman filter implemented on the PX4-based flight controller [13]. The filtered odometry serves as an input to the SLAM module, which, using depth images, constructs an occupancy grid Map of the environment and refines the robot's pose. The RTAB-Map algorithm [14] is employed for this purpose. This software is versatile, supporting various sensors like lidars, stereo cameras, and depth cameras, and it offers the flexibility to utilize external odometry sources or compute odometry using advanced visual or lidar odometry techniques.

C. Planning

To generate obstacle-free paths, the system employs a sample-based planner, specifically the classical rapidly exploring random tree (RRT) star planner [15]. This planner takes a planning time as input and retrieves the best random geometric path within the given time frame. Given the partially known environment in which the drone operates, the path is continuously checked for re-planning during motion. In such cases, a faster planning time is requested to enable the UAV to promptly react to sudden obstacles appearing on its path. The Open Motion planning library (OMPL) [16] is used to implement the planner, offering a convenient way to employ path-planning algorithms with a range of well-known planning approaches. As the environment is assumed to be semi-unknown, ensuring obstacle avoidance is crucial for successful autonomous navigation. The Flexible Collision library (FCL) [17] evaluates the validity of a given state by considering the current configuration of the UAV, its shape, and the environment's state encoded using the OctoMap structure. The OctoMap is constructed by the SLAM algorithm, as explained in Section III-B.

Once the planner finds an obstacle-free path, the trajectory planner generates a quintic polynomial motion trajectory for each position coordinate. This trajectory follows the equation [18]

$$p(t) = a_5 t^5 + a_4 t^4 + a_3 t^3 + a_2 t^2 + a_1 t + a_0 \qquad (1)$$

where $t \in \mathbb{R}$ is the time variable, $p(t) \in \mathbb{R}$ represents a generic position variable, and $a_i \in \mathbb{R}$, with $i = 0, \ldots, 5$, are the polynomial coefficients. The coefficients can be computed by imposing $t = 0$ and $t = t_f$, with $t_f \in \mathbb{R}$ the final time, on the position variable and its first two derivatives. This process is repeated for the three linear movements and the heading trajectory to determine the velocity and acceleration profiles as well.

Hence, the execution of the planning module can be summarized as follows: first, the RRT star planner generates an obstacle-free path, and then the motion planner generates the velocity and acceleration profiles for the drone to follow. Meanwhile, the collision check runs in the background during motion, ensuring that the path remains free from obstacles. If any obstacles are detected along the path, a re-plan message is sent to the path planner to compute a new safe path. To ensure that the drone perceives obstacles along its trajectory, its heading is planned to always align with the direction of the path generated by the RRT star algorithm. This approach is also useful for directing the drone towards inspection points.

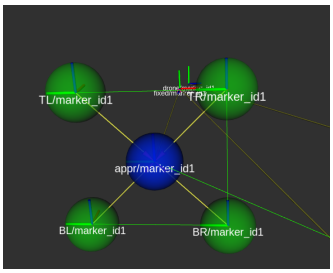


Fig. 3: Inspection targets.

D. Inspection Spot Detection and Targets Generation

During flights in inspected industrial plants, the aerial robot may encounter inspection spots that require more precise exploration, depending on the specific characteristics of the plant being monitored. These spots could include cracks in concrete or pavement structures, corrosion on pipes or steel structures in oil and gas plants, or other surface health issues. However, since this work focuses on the strategy for approaching and inspecting such spots rather than on detecting specific damages, damaged spots are assumed to be replaced with ArUco markers. This allows the focus to remain on the approach and inspection strategy once the spots are detected.

The proposed approach involves two main steps: detection of the spot and generation of inspection viewpoints, followed by execution of the approach and monitoring trajectory. For the first step, an ArUco detector was developed using the OpenCV library [19]. Let $\mathcal{C}, \mathcal{B}$, and $\mathcal{W}$ be the camera, body, and world frames, respectively. This detector returns the pose of the marker in the camera frame $\mathbf{p}^{\mathcal{C}}_{marker} \in \mathbb{R}^3$, with the y-axis pointing upwards and the x-axis pointing rightwards relative to the marker. Notice that we will indicate the homogeneuos representation of a point with $\tilde{\mathbf{p}}^{\mathcal{C}}_{marker} = \begin{bmatrix} \mathbf{p}^{\mathcal{C}^T}_{marker} & 1 \end{bmatrix}^T \in \mathbb{R}^4$. The first viewpoint is then generated to allow the aerial platform to approach the inspection spot, placing it in front of the detected marker with a certain offset that can be adjusted based on the environment being monitored. The approach viewpoint, $\mathbf{p}^{\mathcal{W}}_{appr} \in \mathbb{R}^3$, can be described as follows

$$\tilde{\mathbf{p}}^{\mathcal{W}}_{appr} = \mathbf{T}^{\mathcal{W}}_{\mathcal{B}} \mathbf{T}^{\mathcal{B}}_{\mathcal{C}} \mathbf{T}_{appr} \tilde{\mathbf{p}}^{\mathcal{C}}_{marker} \qquad (2)$$

Where $\tilde{\mathbf{p}}^{\mathcal{W}}_{appr} = \begin{bmatrix} \mathbf{p}^{\mathcal{W}^T}_{appr} & 1 \end{bmatrix}^T \in \mathbb{R}^4$ is the homogeneous representation of the approach point in the world frame. The $\mathbf{T}_{appr} \in SE(3)$ represents a homogeneous transformation that defines the pose of the approach viewpoint computed from the marker pose. It is defined as

$$\mathbf{T}_{appr} = \begin{bmatrix} \mathbf{R}_{appr} & \mathbf{o}_{appr} \\ \mathbf{0}^T & 1 \end{bmatrix}, \qquad (3)$$

in which $\mathbf{R}_{appr} \in SO(3)$ is a rotation matrix that rotates the marker frame, it includes, respectively, a rotation of $\frac{\pi}{2}$ radians around the x-axis and a rotation of $-\frac{\pi}{2}$ radians around the y-axis, these rotations help to align the approach frame with the x-axis pointing towards the marker, i.e. the inspection spot. Then, $\mathbf{o}_{appr} \in \mathbb{R}^3$ is an offset vector that characterizes the distance of the approach point from the inspection spot. In the application of this work, this vector will have just one non-zero value, that will be an offset along the x-axis that will determine the distance from the spot during the inspection. The transformation matrices $\mathbf{T}_{\mathcal{C}}^{\mathcal{B}} \in SE(3)$ and $\mathbf{T}_{\mathcal{B}}^{\mathcal{W}} \in SE(3)$ bring up the approach frame concerning to the world fixed frame $\mathcal{W}$. The former utilizes the extrinsic parameters of the camera, which describe the rotations and translations of the camera relative to the body frame. These parameters are determined during the camera's installation process. Conversely, the latter relies on the transformation between the robot's body frame and the world frame, provided by the localization system. Once the approach viewpoint has been generated, four additional viewpoints are created to draw a square-like inspection trajectory (see Fig. 3)

$$\begin{aligned} \tilde{\mathbf{p}}_{TL}^{\mathcal{W}} &= \mathbf{T}_{off_1} \tilde{\mathbf{p}}_{appr}^{\mathcal{W}}, \\ \tilde{\mathbf{p}}_{TR}^{\mathcal{W}} &= \mathbf{T}_{off_2} \tilde{\mathbf{p}}_{appr}^{\mathcal{W}}, \\ \tilde{\mathbf{p}}_{BR}^{\mathcal{W}} &= \mathbf{T}_{off_3} \tilde{\mathbf{p}}_{appr}^{\mathcal{W}}, \\ \tilde{\mathbf{p}}_{BL}^{\mathcal{W}} &= \mathbf{T}_{off_4} \tilde{\mathbf{p}}_{appr}^{\mathcal{W}}, \end{aligned} \tag{4}$$

where $\mathbf{T}_{off_i} \in SE(3)$ are the offset transformation matrices that locate the inspection viewpoints in the top-left (TL), top-right (TR), bottom-left (BL), and bottom-right (BR) positions. The distance from the approaching target, which determines the size of the squared inspection trajectory, can be adjusted using these transformation matrices. After generating the references, the inspection task is divided into two trajectories: the first trajectory involves approaching the target point $\mathbf{p}_{\text{appr}}^{\mathcal{W}}$ and aligning the drone's heading to point towards the inspection spot. Subsequently, the second trajectory executes the inspection task by moving along the square-like trajectory with the heading fixed toward the marker. The collision checking still runs in the background and checks if the inspection trajectory generated is feasible.

E. Executive System

The execution and supervision of complex activities are facilitated by equipping each team member with an agent-specific executive system, akin to the framework proposed in [20], [21]. This system comprises three fundamental modules: a long-term memory (LTM) for storing task descriptions, a working memory (WM) responsible for maintaining a hierarchical representation of structured tasks during execution, and a behavior-based system (BBS) that provides a symbolic representation of the agent's primitives, including controllers and perceptual modules for execution.

During execution, tasks are retrieved from LTM and allocated to the robots' WM for monitoring and further decomposition until primitive nodes/behaviors are reached.

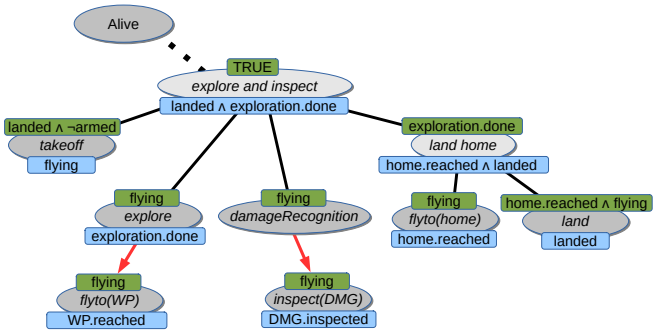


Fig. 4: Representation of the explore and inspect task allocated into the agent's working memory. Light/dark gray ovals are for abstract/concrete nodes, green/blue rectangles are pre- and post-conditions, while red arrows represent the dynamic allocation of nodes.

Each task node can be either abstract or concrete. Abstract nodes represent compound tasks/activities to be decomposed, while concrete nodes are associated with executable primitives from the BBS. Both types of nodes have associated pre-conditions, which determine nodes' activation, and post-conditions, which are satisfied if a node is accomplished. Pre- and post-conditions are represented as conjunctions of binary state variables. A concrete node can only be executed if all parent nodes in the hierarchy are both active and not accomplished. Additionally, active tasks are subject to attentional regulations, which are used to schedule execution when multiple actions are enabled simultaneously.

In our scenario, we assume a hierarchically structured exploration and inspection task to be allocated into the system's WM (see Fig. 4). The task comprises an "exploration" subtask, which dynamically selects waypoints to be reached (`flyto(WP)` in the figure). Once a waypoint is reached

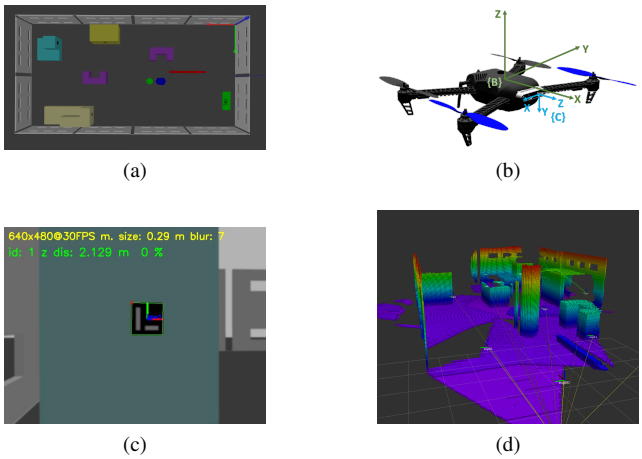


Fig. 5: Simulated experiments: drone, inspection sensor, and world reference frames. Also, obstacles, the first pose of the drone to start the inspection, and the detection of a marker are shown.

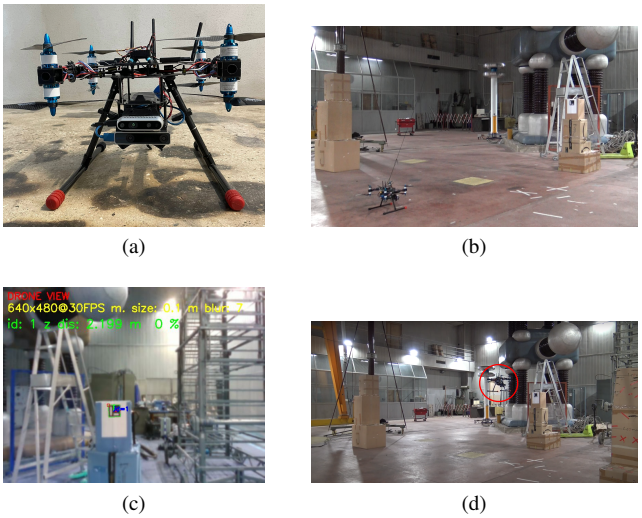
(`WP.reached` is satisfied), the next one is selected for exploration. During exploration, a damage recognition module continually runs, checking for possible damages or defects to be inspected. If a potential damage (DMG) is recognized during exploration, a new inspection task (`inspect(DMG)`) is allocated into the system's WM. Since inspection and exploration activities are not mutually exclusive (i.e., a conflict arises), the agent must decide online which of the two tasks to perform (exploration or inspection). Here, we rely on the attentional regulations of the two tasks to resolve the impasse: the most emphasized activity is selected for current execution, while the other one is postponed. In this scenario, we exploit target proximity as a simple source of emphasis that induces the agent to prioritize nearly accomplished activities.

IV. CASE STUDY

A. Simulated Experiments

To validate the proposed approach, simulated experiments were conducted in a simulated environment created using the Gazebo simulator (Fig.5). The simulated environment consists of a confined space measuring 20×10 m, populated by structured obstacles. Three ArUco markers are distributed over the obstacles, with their poses assumed to be unknown at the start of the inspection operation.

The aerial platform was simulated using the PX4 stack [13] version 1.13.0, specifically the Iris quadcopter equipped with a depth camera. The complete navigation software was implemented using the ROS middleware [22] and using Mavros as a communication layer between the navigation software and the simulated platform. RTAB-Map software was utilized to build the map, generating a 3D OctoMap using the point cloud registered by the depth sensor implementing the SLAM module of Fig. 2. The SLAM algorithm was provided with a simulated odometry message from the Gazebo simulation to mimic real platform modules. Since the executive system was developed in ROS2 [23], the `ros1_bridge` facilitated communication between the ROS and ROS2 systems. The planning module was implemented following the description in Sec. III-C.



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

Fig. 6: Environment in which the actual experiments were conducted. (a) The custom drone was used. (b) The drone in the test arena with cardboard obstacles with markers on it. (c) Onboard view of the detected marker in the approach goal. (d) Drone performing the inspection trajectory in real-world scenario

Assuming a semi-unknown environment, ten exploration points were defined within the perimeter of the simulated plant. The behavior of the aerial platform during the simulation can be observed in Fig. 7, where different commands generated by the executive system are highlighted. At around the sixtieth second, the presence of a marker was detected, initiating the inspection task. As discussed in Sec. III-D, the inspection trajectory consists of two segments represented by differently colored areas. In the top plot of Fig. 7, illustrating the positions of the drone, the square-like trajectory executed after approaching the inspection spot is discernible. In the simulated experiment, the distance from the marker was set to 2 m, while the $\mathbf{T}_{off_i}$ were implemented to define a square inspection trajectory with a side length of 1 m.

B. Real-case Experiments

Real-world experiments were conducted to validate the feasibility of the proposed approach on a real platform. The drone used for this experiment is a coaxial octacopter with a custom carbon fiber frame (see Fig. 6a). It is equipped with an Intel Realsense T265 tracking camera and a Realsense D435i depth camera [1]. The T265 camera features an internal VIO software that computes the odometry of the robot using a stereo fisheye and IMU system onboard an FPGA. The computed odometry is fused with other onboard sensors using the PX4 EKF implementing the Localization Module of Fig. 2. The D435i camera provides point cloud data to the RTAB-Map SLAM algorithm for building the occupancy map. To ensure sensor redundancy and avoid crashes due to poor localization, the drone also mounts a HereFlow optical flow sensor [2], which can be directly connected to the flight controller. The flight controller board used is a PixHawk 6C [3] with PX4 firmware. Additionally, the drone is equipped with a LattePanda 3 Delta [4] as a companion computer.

The test was conducted in a University facility, where a 5×5 m industrial-like environment was defined without GPS signal. ArUco markers with a side size of 0.1 m were placed on cardboard obstacles to simulate inspection spots. The result of a real-world experiment lasting five minutes is illustrated in Fig. 8. Similar to the simulated test, linear position and heading angle are shown. During this test, two markers were applied to the cardboard obstacles ($ID = 1, 2$), while four fixed exploration targets were located to create a square of two meters at a height of

[1] https://www.intelrealsense.com/

[2] https://docs.cubepilot.org/user-guides/flow-senor/here-flow

[3] https://holybro.com/products/pixhawk-6c

[4] https://www.lattepanda.com/lattepanda-3-delta

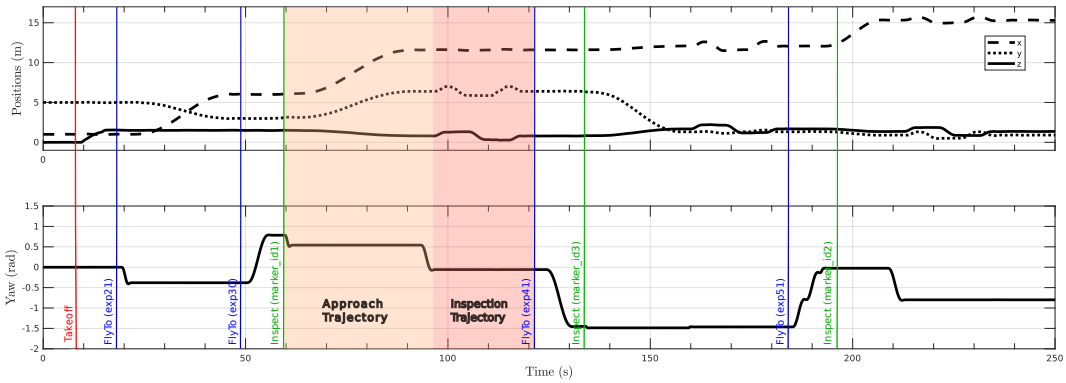


Fig. 7: Simulated case-study result: the top plot illustrates linear positions, while the bottom one shows the yaw of the platform during the task. The actions executed by the platform are illustrated with different colors: red for the take-off, blue for exploration commands, and green for the inspection commands.

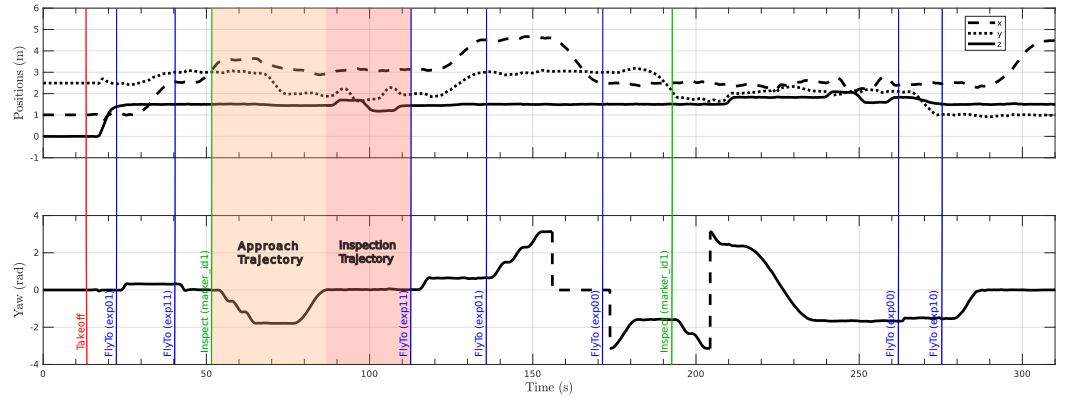


Fig. 8: Real-case experiment result: as for the simulated experiment, the linear position and heading angle of the drone are plotted. The dotted lines in the yaw trajectory replace the well-known discontinuity representation of the Euler angles.

1.5 m. The drone was commanded to cruise at a velocity of 0.3 m/s. In the plot of the result, it can be observed that the drone starts the navigation by exploring the fixed targets. During the trajectory to reach the $exp_{1,1}$ target, it detects the marker with $ID = 1$, interrupts the navigation, generates the inspection viewpoints, and begins the inspection of this spot. Subsequently, the approach trajectory is executed with heading alignment, followed by the execution of the inspection trajectory with a square-like shape. Since the whole framework has to run onboard the drone computer, high relevance could have to consider the total CPU usage. During several tests, the average CPU percentage was 86% and the most computationally heavy process was the ROS manager of the RealSense cameras. This factor, probably, was the reason why sometimes the odometry feedback from one of the RealSense cameras freezes and gives little drift to the aerial robot. This behavior is mitigated by the presence of the optical-flow sensor, which does not require computation to the onboard computer and provides robust feedback to stabilize the drone during localization jumps.

V. CONCLUSION AND FUTURE DIRECTIONS

In this work, a comprehensive autonomous framework for vision-based inspection of industrial plants has been presented. The approach has been thoroughly validated through both simulated and real-world experiments, showcasing its potential application in actual industrial plants. As the focus of the work lies on navigation and the policy adopted to monitor potential inspection spots, there are several avenues for further development to transition this approach into commercial applications.

One potential direction for future development is the integration of defect or corrosion detection capabilities tailored to the specific plant being inspected. Additionally, introducing other agents into the system could be another fruitful area for further enhancement.

REFERENCES

[1] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[2] P. Nooralishahi, C. Ibarra-Castanedo, S. Deane, F. López, S. Pant, M. Genest, N. P. Avdelidis, and X. P. V. Maldague, "Drone-based non-destructive inspection of industrial sites: A review and case studies," *Drones*, vol. 5, no. 4, 2021.

[3] J. A. Tsanakas, L. D. Ha, and F. Al Shakarchi, "Advanced inspection of photovoltaic installations by aerial triangulation and terrestrial geo-referencing of thermal/visual imagery," *Renewable Energy*, vol. 102, pp. 224–233, 2017.

[4] J.-Y. Park, S.-T. Kim, J.-K. Lee, J.-W. Ham, and K.-Y. Oh, "Automatic inspection drone with deep learning-based auto-tracking camera gimbal to detect defects in power lines," in *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, ICVISP 2019, (New York, NY, USA), Association for Computing Machinery, 2020.

[5] F. Mirallès, P. Hamelin, G. Lambert, S. Lavoie, N. Pouliot, M. Montfrond, and S. Montambault, "Linedrone technology: Landing an unmanned aerial vehicle on a power line," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6545–6552, 2018.

[6] R. Carvalho, R. Nascimento, T. D'Angelo, S. Delabrida, A. G. C. Bianchi, R. A. R. Oliveira, H. Azpúrua, and L. G. Uzeda Garcia, "A uav-based framework for semi-automated thermographic inspection of belt conveyors in the mining industry," *Sensors*, vol. 20, no. 8, 2020.

[7] D. Denhof, B. Staar, M. Lütjen, and M. Freitag, "Automatic optical surface inspection of wind turbine rotor blades using convolutional neural networks," *Procedia CIRP*, vol. 81, pp. 1166–1170, 2019. 52nd CIRP Conference on Manufacturing Systems (CMS), Ljubljana, Slovenia, June 12-14, 2019.

[8] K.-W. Tse, R. Pi, Y. Sun, C.-Y. Wen, and Y. Feng, "A novel real-time autonomous crack inspection system based on unmanned aerial vehicles.," *Sensors (Basel)*, vol. 23, mar 2023.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2016.

[10] L. Yu, E. Yang, P. Ren, C. Luo, G. Dobie, D. Gu, and X. Yan, "Inspection robots in oil and gas industry: a review of current solutions and future trends," in *2019 25th International Conference on Automation and Computing (ICAC)*, pp. 1–6, 2019.

[11] S. Roos-Hoefgeest, J. Cacace, V. Scognamiglio, I. Álvarez, R. C. González, F. Ruggiero, and V. Lippiello, "A vision-based approach for unmanned aerial vehicles to track industrial pipes for inspection tasks," in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1183–1190, 2023.

[12] R. B. Sousa, H. M. Sobreira, and A. P. Moreira, "A systematic literature review on long-term localization and mapping for mobile robots," *Journal of Field Robotics*, vol. 40, no. 5, pp. 1245–1322, 2023.

[13] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6235–6240, 2015.

[14] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[15] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt*," in *2011 IEEE International Conference on Robotics and Automation*, pp. 1478–1483, 2011.

[16] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," *IEEE Robotics & Automation Magazine*, vol. 19, pp. 72–82, December 2012. `https://ompl.kavrakilab.org`.

[17] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *2012 IEEE International Conference on Robotics and Automation*, pp. 3859–3866, 2012.

[18] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 2010.

[19] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[20] R. Caccavale and A. Finzi, "Plan execution and attentional regulations for flexible human-robot interaction," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2453–2458, 2015.

[21] R. Caccavale and A. Finzi, "A robotic cognitive control framework for collaborative task execution and learning," *Topics in Cognitive Science*, vol. 14, no. 2, pp. 327–343, 2022.

[22] L. Joseph and J. Cacace, *Mastering ROS for Robotics Programming - Second Edition: Design, Build, and Simulate Complex Robots Using the Robot Operating System*. Packt Publishing, 2nd ed., 2018.

[23] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.