



Designing a Human-Drone Interaction: Insights from the AeroAssistant Framework

Franceschini, Riccardo

Publication date:
2024

Document Version
Publisher's PDF, also known as Version of record

[Link back to DTU Orbit](#)

Citation (APA):
Franceschini, R. (2024). *Designing a Human-Drone Interaction: Insights from the AeroAssistant Framework*. Technical University of Denmark.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Doctor of Philosophy
Doctoral thesis in Photonics Engineering

DTU Electro
Department of Photonics Engineering

Designing a Human-Drone Interaction: Insights from the AeroAssistant Framework

Author: Riccardo Franceschini



Supervisor: Prof. Matteo Fumagalli

Co-supervisors: Prof. Ole Ravn & Prof. Julián Cayero Becerra

DTU Electro
Department of Photonics Engineering
Technical University of Denmark
Ørsted's Plads
Building 340
2800 Kongens Lyngby, Denmark

Summary

The need for efficient infrastructure inspection has grown due to significant annual investments required for maintenance of the aging infrastructure across Europe. Traditional inspection methods are not only time-consuming but also hazardous, underscoring the necessity for faster robotic solutions that can perform these tasks more safely and efficiently. Unmanned Aerial Vehicles (UAVs) have emerged as versatile tools in this domain, but their widespread adoption hinges on the ease and efficiency of their operation where teloperation serves as de-facto the standard. For this reason the thesis introduces AeroAssistant, a framework designed to improve the teleoperation of UAVs through a combination of shared autonomy and augmented reality elements to ease the piloting enabling even non-expert pilot to control the UAV with confidence.

Thus, the research addresses the problem of delivering an effective human drone interaction for aerial inspection from different aspects, creating in such way the building blocks that will be used by AeroAssistant to provide a comprehensive enhanced teleoperation experience. First, a novel method for efficiently retrieving a path capable of optimizing a combination of environmental factors, which could range from obstacle distance and terrain structure to environmental conditions, is proposed. Then, the research shifted to proposing a UAV interaction paradigm for aligning with and following any surface geometry using a convenient interface that requires minimal user intervention. Subsequently, the effectiveness of a collaborative navigation experience was studied along with user evaluation, where the operator is only responsible for defining the direction and velocity of the UAV while the proposed solution ensures collision-free navigation. Lastly, the latest advancements in promptable segmentation models are exploited to create an interaction in which the operator efficiently segments and inspects specific areas of interest, extracting a path capable of covering the entire inspection area with just few clicks.

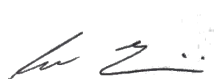
Finally, AeroAssistant is presented. An in depth analysis of the core of the framework is provided, highlighting its architecture and how the different features proposed in the previous papers are integrated as plugins along with other features. For each plugin, an explanation of the integration is provided with both the user interaction paradigm and the operator interface. The underlying features necessary for developing an effective augmented reality interaction are then proposed along with field

experiments and ongoing research.

Preface

The purpose of this thesis is to elucidate the PhD project and its corresponding results. The PhD thesis, entitled "Designing a Human-Drone Interaction: Insights from the AeroAssistant Framework," was conducted from July 2021 to June 2024 under the industrial PhD scheme at Eurecat: Technology Center of Catalunya, located in Cerdanyola del Vallès, Barcelona, Spain. This research was undertaken under the supervision of Matteo Fumagalli, Ole Ravn, and Julian Cayero Becerra. The project was funded by a Marie Skłodowska-Curie PhD scholarship as part of the European project Aero-Train (Marie Skłodowska-Curie Grant Agreement No. 953454). The primary objective of this PhD was to advance the Human-Drone interaction experience within the context of aerial infrastructure inspection. The outcome of the research is the AeroAssistant, a framework designed to facilitate intuitive and efficient interaction between human operators and drones, ensuring safety, reliability, and ease of use in various operational scenarios. The research conducted has led to the publication of four conference papers, one workshop, and the submission of one journal article, showcasing the innovative findings and contributions made during this PhD journey. These publications reflect the rigorous scientific research and practical implications of the work, aiming to influence future developments in the field of Human-Drone interaction.

Kongens Lyngby, 27th June 2024



Riccardo Franceschini

Acknowledgements

I would like to thank my advisor, Prof. Julian Cayero Becerra, and Prof. Matteo Fumagalli for selecting me as a PhD student and giving me the opportunity and freedom to pursue many interesting ideas. I would also like to extend my gratitude to all my colleagues at Eurecat and the Early Stage Researchers in the Aero-Train project for their invaluable support throughout this journey, as well as for the personal connections that goes beyond the work environment. Last but not least, I would like to thank my girlfriend Ginevra, my family, and all my friends for their unwavering constant support over the past three years.

List of publications

- Conference Paper : **R. Franceschini**, M. Fumagalli and J. Cayero Becerra, "Learn to efficiently exploit cost maps by combining RRT* with Reinforcement Learning," 2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Sevilla, Spain, 2022 [50]
- Conference paper : **R. Franceschini**, M. Fumagalli and J. Cayero Becerra, "Enhancing Human-Drone Interaction with Human-Meaningful Visual Feedback and Shared-Control Strategies," 2023 International Conference on Unmanned Aircraft Systems (ICUAS), Warsaw, Poland, 2023, pp. 1162-1167 [54]
- Workshop: **R. Franceschini** , M. Fumagalli, J. Cayero Becerra, Riding the Rollercoaster: easing UAV Piloting Experience with XR and continuous planning (XR-ROB 2023 - Second International Workshop on "Horizons of an Extended Robotics Reality" @ IEEE/RSJ IROS 2023) [55]
- Conference Paper: **R. Franceschini**, J. Rodriguez Marquez, M. Fumagalli and J. Cayero Becerra, "Riding the Rollercoaster: Improving UAV Piloting Skills with Augmented Visualization and Collaborative Planning" 2024 International Conference on Unmanned Aircraft Systems (ICUAS), Chania, Greece, 2024 [56]
- Conference Paper : **R. Franceschini**, J. Rodriguez Marquez, M. Fumagalli, and J. Cayero Becerra, "Point, Segment, and Inspect: Leveraging Promptable Segmentation Models for Semi-Autonomous Aerial Inspection": 33rd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Pasadena, California, USA [79]
- Journal Paper: **R. Franceschini**, M. Fumagalli and J. C. Becerra, "AeroAssistant : a modern and flexible UAV teleoperation framework", submitted at ACM Transaction on Human-Robot-Interaction [69]

Abbreviations

HMD	Head Mounted Display
UAV	Unmanned Aerial Vehicle
MAVs	Micro Aerial Vehicles
OMAVs	Omnidirectional Micro Aerial Vehicles
ROS	Robot Operating System
RC	Remote Controller
POV	Point of View
FPV	First Person View
AA	AeroAssistant
AR	Augmented Reality
VR	Virtual Reality
MR	Mixed Reality
XR	Extended Reality
BVLOS	Beyond Line of Sight
QP	Quadratic Program
SFC	Safe Flight Corridor
SoC	System on a Chip
ARM	Advanced RISC Machine
TOPS	Tera Operations per Second
NAS	Neural Architecture Search
LLMs	Large Language Models

ONNX	Open Neural Network Exchange
SAM	Segment Anything Model
RRT	Rapidly Exploring Random Trees
PPO	Proximal Policy Optimization
SD	Standard Deviation
p_r	Robot Position
q_r	Robot Orientation
r_g	Rollercoaster Gain
r_d	Rollercoaster Direction
p_i	Point of Interest
n_i	Normal Point of Interest
p_c	Closest Point
p_m	Point on image plane
p_d	Point of destination
o_d	Orientation of destination
S	Robot States
s_r	Robot Status
K	Camera Matrix

Contents

Summary	i
Preface	iii
Acknowledgements	v
List of publications	vii
Abbreviations	ix
Contents	xi
1 Introduction	1
1.1 Problem Statement	1
1.2 Research Objectives	3
1.3 AR,VR,MR & XR	4
1.4 State of the art	5
1.5 Summary	10
2 Contribution	11
2.1 Efficient Cost-Aware Path Planning	11
2.2 Collaborative Displacement Along Surfaces	13
2.3 Collaborative navigation with the Rollercoaster	15
2.4 Point Segment and Inspect	18
2.5 AeroAssistant	22
3 AeroAssistant Core	23
3.1 Robot Status	24
3.2 Middleware	25
3.3 Remote Control	25
3.4 Manager Architectures	26
3.4.1 Passive Managers	27
3.4.2 Active Managers	29
3.5 Plugins	30

3.5.1	Lock to a point	31
3.5.2	Move to a point	33
3.5.3	Align and Follow Surface	34
3.5.4	Rollercoaster	36
3.5.5	Point to Segment to Plan	37
3.6	Augmented Visualization	39
3.6.1	Camera Model	39
3.6.2	From Virtual to Real	41
3.6.3	Interacting While Flying	42
3.7	Running on Constrained devices	43
3.8	Field Experiment	46
3.9	Bringing the Pilot into MR headsets	49
4	Conclusion	53
Conclusion		53
4.1	Future Directions	53
4.1.1	Swarm Management	54
4.1.2	Immersive Interfaces	54
4.1.3	Assisted Manipulation	54
4.1.4	Natural Language Interaction	55
Appendices		57
Appendix A Articles		59
Bibliography		111

CHAPTER 1

Introduction

Unmanned Aerial Vehicles (UAVs), thanks to their ability to move freely in space, have emerged as versatile tools with applications ranging from aerial surveillance and infrastructure inspection to disaster response. However, their widespread adoption often relies on the ease and efficiency of their teleoperation as the human supervision remain fundamental in ensuring safety. In this context, this thesis proposes AeroAssistant, an innovative assistive UAV teleoperation framework designed to transform the user experience for both expert and novice pilots. AeroAssistant combines shared autonomy and augmented reality elements to simplify UAV operation, overcoming traditional limitations and empowering operators with tools to perform intricate tasks with precision and confidence.

Throughout this thesis, we explore the conceptual foundations of shared autonomy and augmented reality in UAV teleoperation, as well as their practical implementation within the AeroAssistant framework. Structured as a collection of papers, this thesis explores the conceptualization, development, and evaluation of AeroAssistant, offering insights into its design principles, functionalities, and applications. The subsequent sections discuss the motivation behind this work (Sec. 1.1), the objectives (Sec. 1.2), the distinctions among various paradigms of immersive visualization (Sec. 1.3), and analyze the state of the art in teleoperation and aerial robot interaction (Sec. 1.4). In Chapter 2, a summary of the papers included in Appendix A is provided, while Chapter 3 focuses on the core approach, analyzing the adjustments necessary to achieve effective human-drone interaction, showcasing field experiments, and ongoing investigations. Finally, in Chapter 4, conclusions are drawn, and future research directions are proposed.

1.1 Problem Statement

In recent years, there has been an increasing demand for the efficient assessment of the condition of various components of civil infrastructure. According to projections from the Organization for Economic Cooperation and Development (OECD), Europe is expected to allocate a substantial €1.2 trillion annually between now and 2030 to ensure the ongoing maintenance of its aging infrastructure [1]. This encompasses critical elements such as roads [2], bridges [3,4], power lines [5], solar plants [6], and wind

turbines [7], among others. The importance of this assessment cannot be overstated, particularly considering the substantial investments required for maintenance and upkeep. Inspections typically involve a broad spectrum of activities, ranging from visually identifying defects like cracks or corrosion (Figure 1.1(b)) to non-destructive testing (NDT). In NDT, sensors (e.g., ultrasonic sensors, Figure 1.1(a)) come into contact with surfaces, such as tanks in oil & gas facilities or the lightning protection systems (LPS) in wind turbines, to analyze potential structural defects.

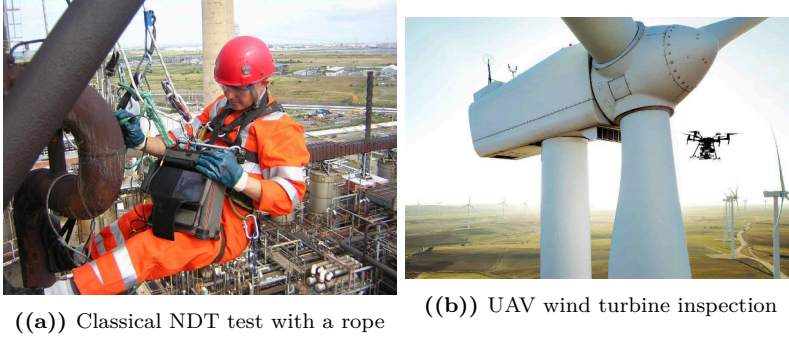


Figure 1.1. Different inspection situations

These operations are inherently time-consuming due to the extensive footprint of the facility being inspected and risky as to perform such inspections may require erecting scaffolding or using ropes. Hence, automation and robotization of such tasks present viable solutions by eliminating human involvement in hazardous operations, while also enabling the deployment of sensors and tools to perform intricate activities in remote and difficult-to-access environment.

Due to their ability to navigate freely and access spatially challenging areas, UAVs are increasingly preferred over traditional inspection methods, thanks to their capacity for customization with specialized sensors and tools tailored for specific tasks. However, given the safety-critical nature of the environments where these inspections occur, such as oil&gas facilities, power lines or nuclear plants, human pilots are still required to oversee and control the operations. One of the main problem is that, operating in these environments presents a significant challenge, even for experienced pilots, who may have to control the UAV far from their line-of-sight, limiting the overall situational awareness in cluttered and hazardous scenarios (Figure 1.2).

Within the world of professional pilots, a study by [8] interviewed pilots about their perspectives on human-drone interaction, focusing on what matters most when piloting UAVs. The study highlighted that safety is the most relevant priority for the pilots and establishing clear communication between the drone and the operator is a primary concern for them. This emphasis on safety underscores the importance of improving trust between the operator and the technology [9], ultimately encouraging the operator to utilize features with confidence in the drone's actions.



Figure 1.2. Pilot and inspector performing blade inspection

In light of these insights, particularly in operational contexts, there is a growing anticipation for a form of robot interaction where the operator can guide the UAV at a high level, issuing commands such as inspecting specific surfaces or maintaining distance and alignment with particular targets, all while retaining control of the overall situation. This concept aligns with the framework proposed in [10], which categorizes different levels of autonomy in human-robot interactions. The desired interaction paradigm in such scenarios is identified as "Shared Control With Human Initiative" wherein the operator maintains decision-making authority while the UAV executes actions with minimal oversight.

Incorporating mechanisms like these, alongside augmented interfaces as demonstrated in [11], has proven effective in enhancing user acceptance through augmented visualization, becoming pivotal in establishing efficient interaction mechanisms.

1.2 Research Objectives

As discussed previously in section Sec. 1.1, the aim is to develop improved interaction methods between the UAVs and the operators, with the idea of simplifying the pilot's experience decreasing their mental load while ensuring they still feel in control. To achieve this goal, this thesis research is centered on two interconnected aspects: shared autonomy routines and augmented reality interfaces.

The first aspect involves creating customizable shared interaction schemes. These schemes provide autonomous and semi-autonomous routines that operators can trigger and control during flight.

The second aspect, augmented reality interfaces, addresses the need to establish trust between the operator and the UAV. Each shared autonomy routine developed is accompanied by a specific visualization to communicate UAV intentions and actions. This blend of enhanced visualization and cooperation is integrated into a teleoperation framework called AeroAssistant. AeroAssistant features an internal modular architecture that enables the implementation of various enhanced shared autonomy

algorithms. Additionally, it proposes interaction patterns with a remote controller to offer a familiar interaction scheme.

The ultimate research objective is to develop a system that enables both expert and non-expert pilots to conduct complex UAV operations confidently and efficiently. This objective aims to address the research question: "How can we enhance the effectiveness and intuitiveness of modern UAV systems?"

1.3 AR,VR,MR & XR

Before entering into the domain of aerial teleoperation, where augmented reality (AR) and virtual reality (VR) serve as crucial tools, it's prudent to clarify the terminology, especially for those less familiar with these concepts. Augmented Reality (AR) enriches the real-world environment by superimposing digital information onto it, enhancing our perception and interaction with the physical world. Conversely, Virtual Reality (VR) immerses users in entirely synthetic environments, effectively disconnecting them from their surroundings and transporting them to a virtual world. Mixed Reality (MR), a more subtle concept, blends the virtual and physical worlds seamlessly, allowing digital and real-world elements to coexist and interact in real time. MR offers a spectrum of experiences, from fully immersive digital overlays on the physical environment to partial digital enhancements. Extending beyond these individual terms, Extended Reality (XR) serves as an overarching category, encompassing AR, VR, MR, and potential future developments (Figure 1.3). During the thesis, AR and VR will be analyzed as valid tools for effective aerial interaction. However, the proposed work focuses on AR, showcasing novel interaction paradigms that are further enhanced by AR visualization.

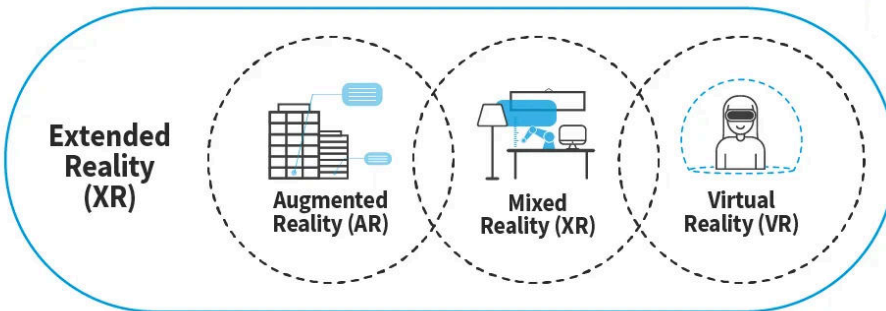


Figure 1.3. Visual representation of the different realities [12]

1.4 State of the art

The landscape of human-robot interaction, especially in teleoperation scenarios, has seen a variety of approaches aimed at improving user experience and system performance. An in depth review of the numerous approaches has been carried out in [13] where the authors have categorized the various interaction schemes that have been explored for fostering human robot interaction. The categorization thought comprehend all the possible interaction patterns with robots including those methods not suitable for human drone interaction. Thus a more dedicated investigation of different interaction schemes developed is here proposed. Numerous studies have explored various strategies, such as XR interfaces, shared control mechanisms employing haptic devices, integration of deep learning agents, as well as the utilization of path planning and collision avoidance systems. Specifically focusing on drone interaction, these approaches can be categorized broadly in different high-level categories.

Assisted Standard Pilot Interfaces: The first category encompasses systems akin to those discussed in [14–16], where authors have investigated the impact of an enhanced teleoperation interaction, resembling standard interaction with the operator controlling the UAV's velocity. However, in this case a collision avoidance algorithm is implemented between the operator and the UAV to adjust the operator's commands if they pose a collision risk (Figure 1.4). Another viable interaction is proposed in [17]. In this work, the authors introduce the concept of teleoperation constrained to predefined virtual surfaces. In such interaction, the operator defines the structure of the surface, which can be a wall or a cylinder depending on the use case. The UAV is then constrained to move along these surfaces, with the operator controlling the movement along them.

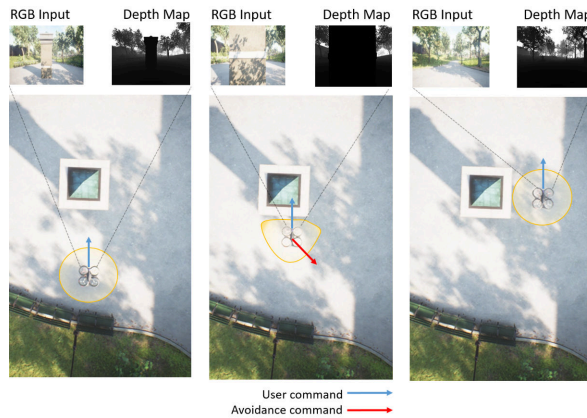


Figure 1.4. Collision avoidance with [16]

Interfaces for Point and Move: Studies such as those elaborated in [18–24] introduce interfaces tailored for controlling nearly autonomous drones. These methods typically require the use of tablets or head-mounted displays (HMDs), with interaction focused on visualizing 3D data collected by the drone or the drone’s trajectory overlaid on a physical model in front of the operator (Figure 1.5). The primary interaction involves defining a destination point or an entire plan and visualizing the UAV movement in space through the proposed interfaces, which depict a miniaturized world. The main drawback of those approaches is that does not account for the possibility of the operator to temporarily gaining control in case of failure or perform minor adjustments in the trajectory.

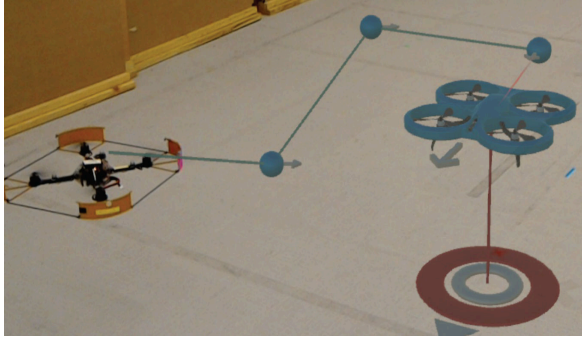


Figure 1.5. Virtual Surrogate by [19]

Haptic Control for Trajectory Manipulation: Alternative approaches, as examined in [25–30] (Figure 1.6), concentrate on employing haptic devices to adjust trajectories, enriching control experiences by offering diverse haptic feedback schemes to users. These methods engage the operator in the UAV control process, empowering them to execute vertical and lateral displacements along a predefined trajectory while receiving haptic feedback concerning obstacle and platform constraints. The incorporation of haptics in teleoperation is particularly pertinent because through non-visual cues, the operator gains immediate perception of potential platform limitations or surrounding obstacles based on the proposed implementation.

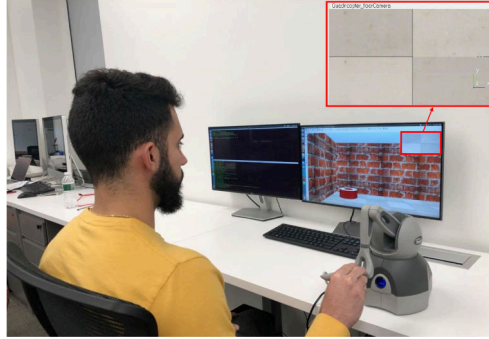


Figure 1.6. Haptic Teleoperation from [30]

Deep Learning Assistance: Interesting teleoperation methodologies are proposed in [31–33] (Figure 1.7). Here, the authors investigate the utilization of learning algorithms to anticipate common trajectories, such as turns, straight lines, or circular shapes, by interpreting the operator’s control inputs from a standard joypad. An agent is positioned between the operator and the UAV, assuming control when it identifies the operator’s attempt to execute one of these trajectories, thereby executing precise movements. This approach is particularly intriguing as the operator continues with a familiar teleoperation approach in which they are confident, while in the background, an autonomous agent runs and takes control only when a learned trajectory is recognized, rendering its actions invisible to the operator.

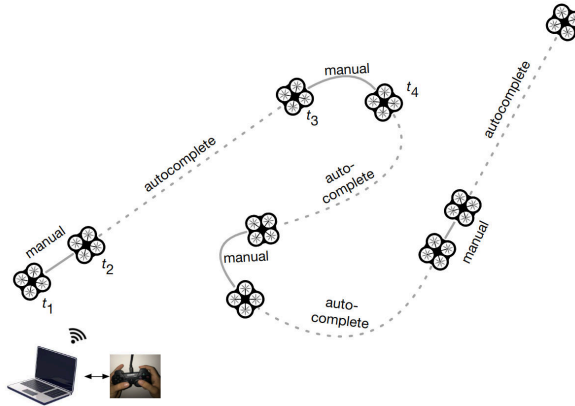
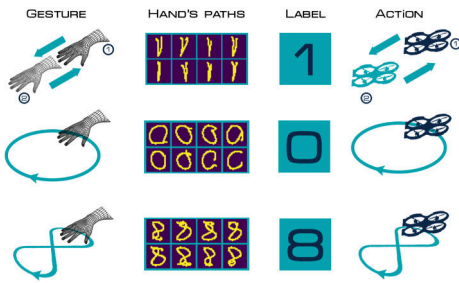


Figure 1.7. Autocomplete interaction concept [31]

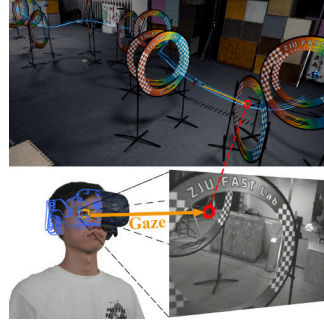
Gaze and Body Control Navigation: Other approaches have also explored natural interaction through gaze and body gestures to achieve a more intuitive experience. Works such as [34–37] have investigated the possibility of controlling UAVs through

body movements (Figure 1.8(a)). Some utilize embedded devices like smart gloves, while others employ computer vision algorithms to detect specific movements. These interactions rely on mapping high-level commands to actions such as takeoff, landing, or executing certain trajectories like circular, lateral, or vertical displacements. Controlling the UAV with these approaches implicitly requires delegating finer control to the UAV itself, as they rely on sending high-level commands. Notable commercial implementations of body gestures are proposed in [38, 39], where semi-autonomous drones are controlled through gestures for tasks like capturing images or executing specific shots.

Moreover, intriguing interaction methods are introduced in [40, 41], where the operator's gaze is detected via a Head-Mounted Display (HMD) and utilized to discern the operator's intentions (Figure 1.8(b)), adjust the UAV trajectory, or designate a point for UAV movement. These approaches are particularly interesting because the eyes serve as a precise mechanism for indicating intention, and they can be integrated with other interaction mechanisms such as standard RC controls.



((a)) Body gesture control from [35]



((b)) Gaze intention control [40]

Figure 1.8. Different Gaze and Body control interactions

AR-assisted navigation: Several studies have explored the use of AR to enhance the navigation experience. For instance, in [41, 42], the authors investigated the impact of AR as a blending mechanism between the operator's perspective and the UAV's perspective (Figure 1.9(a)). They focused on visualizing what the UAV perceives on the operator's headset, integrating this information with the operator's real-world environment accurately. This integration provides immediate feedback to the operator regarding the drone's visual perception and its spatial localization, thereby enhancing operator situational awareness. On the other hand, research such as [43] propose to augment the camera with spatial elements to highlight possible points of interest or insert artificial obstacles, thereby combining virtual and real-world approaches. Additionally, works like [44] proposed an AR-based bird's-eye view, enabling the operator to view the drone from a third-person perspective, en-

hancing spatial understanding of the surrounding environment.

Meanwhile, other studies have explored the usage of AR interfaces to visualize, control, and plan different configurations of omnidirectional micro aerial vehicles (OMAVs) in space, as seen in [45,46]. These OMAVs, due to tilting propellers, offer an intriguing solution by generating thrust in any direction, thus enabling hovering and reaching arbitrary orientations in space (Figure 1.9(b)).

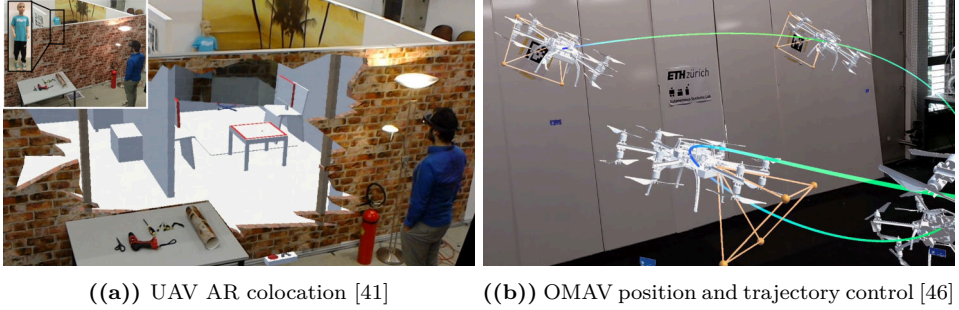


Figure 1.9. Different AR interaction approaches

VR Teleoperation: Another approach involves VR teleoperation interfaces, with a focus on aerial manipulation, as explored in [47, 48]. VR control is particularly relevant as it allows operators to embody themselves inside the UAV, resulting in a high level of spatial awareness for performing manipulation activities depending on the integrated effector (Figure 1.10). Additionally, more commercially oriented approaches utilize VR headsets for controlling FPV drones [49]. In this scenario, the operator experiences the UAV’s first-person view, ensuring maximum awareness, while retaining full control of the drone through an RC controller.



Figure 1.10. AeroVr Interaction [47]

1.5 Summary

In summary, this chapter serves as a general introduction to the topics covered in the thesis. It begins by introducing the problem of aerial inspection, highlighting why UAVs are key components in performing inspection and maintenance operations efficiently and safely. From Sec. 1.4, an overview of the current human-drone interaction paradigms is given, showcasing the efforts of the research community in proposing innovative teleoperation experiences while often focusing on narrow aspects of human-drone interaction. This highlights the need for a comprehensive and customizable human-drone interaction framework capable of combining AR elements with a shared-interaction algorithm to propose a familiar yet enhanced teleoperation experience.

The research conducted during the PhD, detailed in the subsequent chapters, focuses on proposing algorithms and visualization schemes aimed at simplifying common tasks in UAV teleoperation, particularly in the context of aerial inspection. This is achieved without compromising the operator's sense of control. Additionally, the proposed features are integrated into a general yet efficient framework named AeroAssistant, designed from the ground up with efficiency and flexibility in mind, allowing room for future expansion of both the shared-interaction algorithm and augmented visualization.

CHAPTER 2

Contribution

This chapter summarizes the main contributions of the research conducted, utilizing the published results which are then reprinted in the Appendix A. The scientific production comprises 4 published conference papers, 1 workshop paper and 1 journal article which has been submitted and is currently under the review process. The conducted research encompasses a range of topics, from intelligent path planning techniques capable of adapting to various cost-maps, to the implementation and evaluation of advanced shared autonomy algorithms. These algorithms serve as foundational building blocks within the AeroAssistant architecture. The following sections are then summarizing the published works in temporal order.

2.1 Efficient Cost-Aware Path Planning

Motivated by the fact that safety is among the biggest concerns of UAV pilots, the first work published at the 2022 IEEE International Symposium on Safety, Security, and Rescue Robotics is:

(P1:) *R. Franceschini, M. Fumagalli and J. Cayero Becerra, "Learn to efficiently exploit cost maps by combining RRT* with Reinforcement Learning," 2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Sevilla, Spain, 2022 [50]*

This work aims to address the problem of efficiently exploiting cost-maps by proposing RL-RRT*, a sample-efficient and cost-aware sampling method based on RRT* [51]. The approach leverages a deep-learning agent which receives as input the information of the cost maps from both a global and local perspective. Cost maps can represent distances to obstacles, as well as distances to the goal or environmental factors such as elevation or wind conditions if known. The information from the cost maps is then extracted and used to guide the sampling process in subportions of the map, following a standard RRT* sampling process. After sampling n points within the confined region, the network is queried again to redirect the sampling area. A representation of the architecture is provided in Figure 2.1, while the algorithm's pseudocode is given in Algorithm 1.

The method was trained using a reinforcement learning approach employing Proximal Policy Optimization (PPO) [52] due to its data efficiency and reliability. The

Algorithm 1 Environment Step

```

1: procedure ENV.STEP(action)
2:   % get the sampling area from the agent action
3:   boundaries  $\leftarrow$  env.get_sampling_bound(action)
4:   % perform n step of RRT* within the boundaries
5:   state, reward, done  $\leftarrow$  env.plan_n_step(boundaries)
6:   % return the information to the agent
7:   return state, reward, done
8: end procedure
  
```

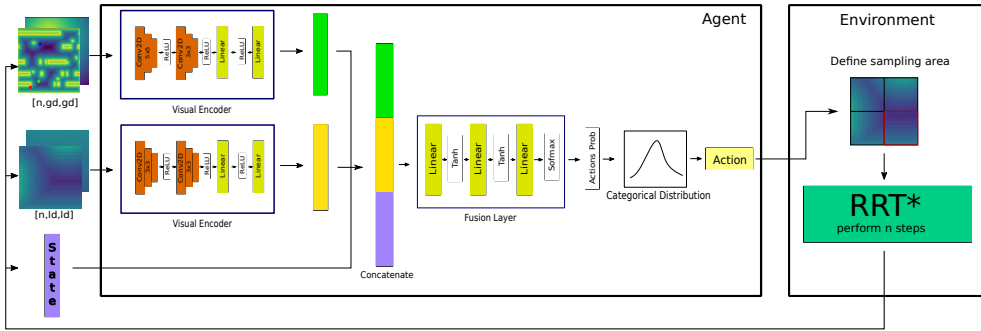


Figure 2.1. The RL-RRT* architecture. The agent receive as input state of the planner plus global and local maps. Using this information the agent drives the RRT* sampling by providing one of the 4 directions, then n nodes are sampled and the loop is repeated until a solution or the maximum iteration is reached.

simulation environment comprised of 2D randomly generated maps, and the agent’s objective was to balance the combined cost of the path along with the number of iterations necessary to find a path to the goal. Consequently, the agent learned to efficiently exploit the costmaps, finding viable paths to the goal in fewer iterations and with lower cumulative costs over the path. Subsequently, the method was compared against RRT*, a sampling-biased version of RRT* where parts of the map with lower costs were more likely to be sampled, and T-RRT* [53], a method that incorporates cost maps into its sampling strategy by accepting or rejecting new nodes based on their cost variation. A visual comparison of the retrieved paths is presented in Figure 2.2, demonstrating the ability of the proposed method to limit the sampling process to only the necessary region while balancing the cost map, represented in this case by the obstacle distance.

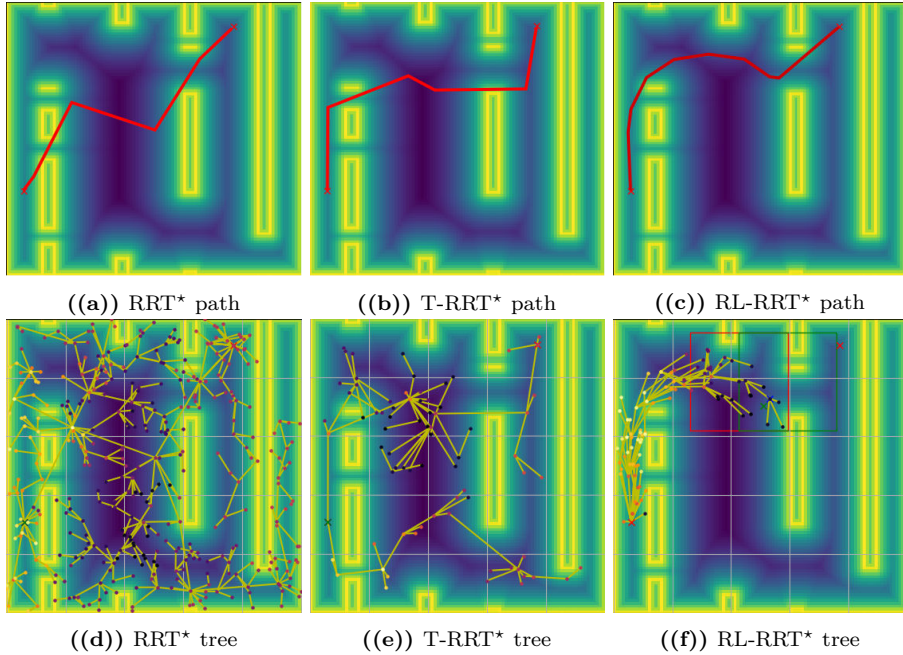


Figure 2.2. Path retrieval comparison between the different approaches

2.2 Collaborative Displacement Along Surfaces

The second proposed work is motivated by the often encountered need during inspections to maintain a certain distance and orientation with respect to the inspected surface. However, controlling the UAV in such a manner is particularly challenging, especially when operations are conducted far from the operator or beyond line of sight (BVLOS), where spatial understanding is limited and weather conditions may be difficult to perceive and counteract. Thus, a system to address this challenge is proposed in:

(P2) *R. Franceschini, M. Fumagalli and J. Cayero Becerra, "Enhancing Human-Drone Interaction with Human-Meaningful Visual Feedback and Shared-Control Strategies," 2023 International Conference on Unmanned Aircraft Systems (ICUAS), Warsaw, Poland, 2023, pp. 1162-1167, doi: 10.1109/ICUAS57906.2023.10156190. [54]*

The proposed work aims to create an interaction paradigm in which the operator receives enhanced camera feedback represented in an AR fashion, providing information regarding alignment with the facing surfaces and the current closest point (see Figure 2.3).

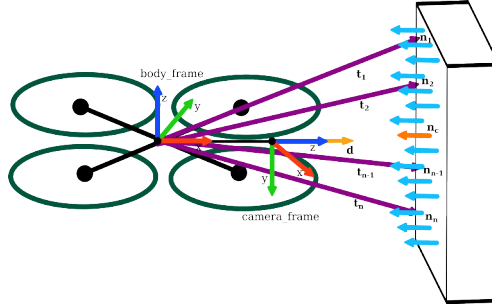


Figure 2.3. Representation of UAV reference frames and vectors used to retrieve the alignment information

The operator is then presented with a user-friendly interface for autonomously aligning the UAV with the nearest surface and executing lateral and vertical displacements while preserving orientation and distance. The overall architecture representing the proposed interaction is depicted in Figure 2.4.

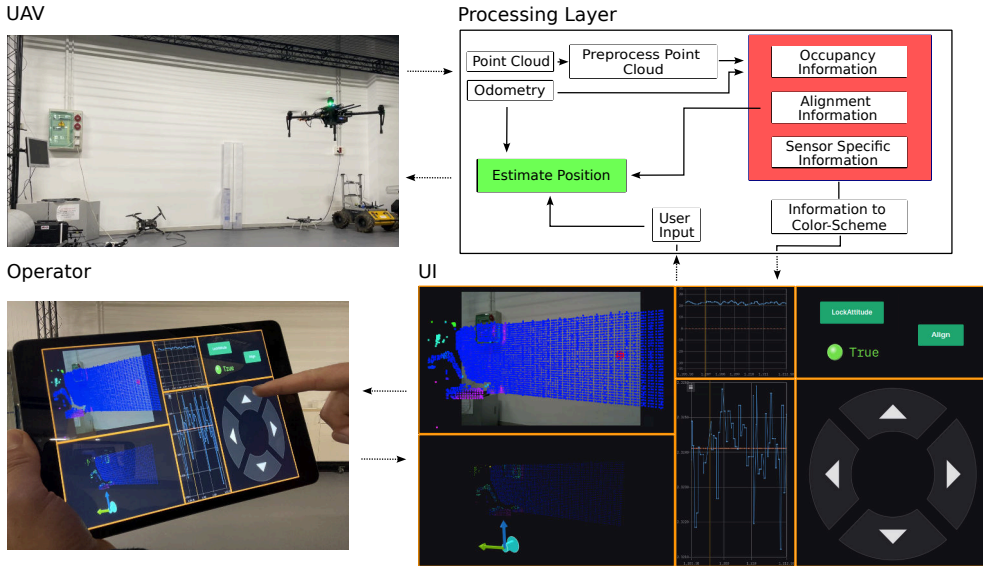


Figure 2.4. The schema depicts the interaction between the operator and the UAV through a tablet interface. Video feedback from the UAV is shown in the upper left corner, with surface points colored according to UAV alignment. The bottom left panel provides a 3D visualization of the UAV reference frame and colored point cloud data.

The visualization provided to the operator offers immediate understanding of the geometric structure of the facing surface. Based on the normal orientation, a heat-map-based color representation is overlaid onto the camera stream. A comparison

between a situation where the UAV is aligned to the surface is presented in Figure 2.5. In Figure 2.5(a) the UAV is aligned and the closest point is reported in green, along with the normal heatmap highlighting the aligned points with a brighter color scheme. In contrast, Figure 2.5(b) shows the closest point reported in red, indicating misalignment with the surface, and the heatmap representation has a darker color scheme. Thus the operator can lock the position and orientation of the drone with respect to any surface and control vertical or lateral displacement and the magnitude of the movement with a convenient pad over the graphical interface.

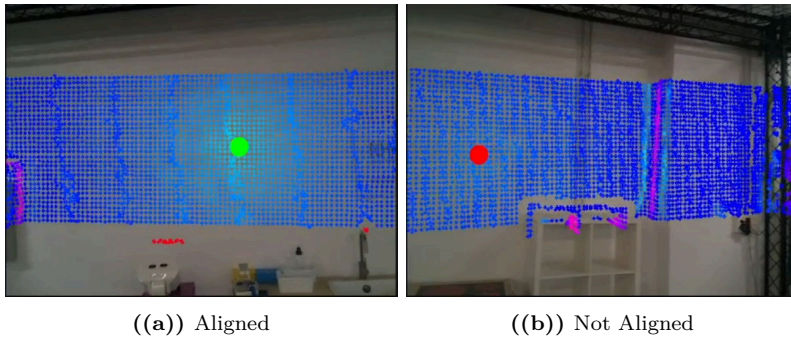


Figure 2.5. Alignment visualization under two different circumstances

2.3 Collaborative navigation with the Rollercoaster

The third work aims to address the challenge of intuitive and safe navigation in any environment by establishing a collaborative navigation paradigm. In this paradigm, the operator controls the direction and velocity at which the UAV should move, while a planner ensures safe navigation. Throughout the navigation process, the UAV maintains the operator aware through an enhanced interface. The primary goal of this proposed solution is to improve the operator’s situational awareness, perception, as well as the safety and efficiency of UAV navigation.

(W3) *Riccardo Franceschini, Matteo Fumagalli, Julian Cayero Becerra, Riding the Rollercoaster: easing UAV Piloting Experience with XR and continuous planning (XR-ROB 2023 - Second International Workshop on "Horizons of an Extended Robotics Reality" @ IEEE/RSJ IROS 2023)* [55]

(P3) *Riding the Rollercoaster: Improving UAV Piloting Skills with Augmented Visualization and Collaborative Planning": Riccardo Franceschini, Javier Rodriguez Marquez, Matteo Fumagalli, and Julian Cayero Becerra, ICUAS 2024* [56]

This work integrates a planner [57] that guarantees real-time safe path retrieval through quadratic programming (QP) featuring the Safe Flight Corridor (SFC) con-

cept. The SFC comprises a collection of convex, overlapping polyhedra that model the available free space and establish a continuous path from the robot's current position to the goal destination. By incorporating the SFC, a series of linear inequality constraints are introduced into the QP, thereby enabling real-time path planning. This path retrieval is then coupled with the mapping method proposed in [58], which utilizes a sliding window approach for planning in local space, thereby circumventing the complexities and drift associated with global approaches. The operator then interacts with the system using a Sony DualSense™ Controller [59]. By pressing the left trigger, the Rollercoaster is activated, and adjusting the pressure a parameter $r_g \in [0, 1]$ called rollercoaster gain is used to linearly scale the velocity along the path according to the the maximum allowed velocity by the platform. Using the right thumbstick, the operator steers the temporal destination point \mathbf{p}_d , which is utilized by the planning algorithm for continuous retrieval of a safe path. This point is positioned at a fixed distance of k m in front of the UAV, and the direction in which \mathbf{p}_d is placed relative to the UAV position is controlled by a parameter $r_d \in [-1, 1]$, known as the rollercoaster direction, as illustrated in Figure 2.6.

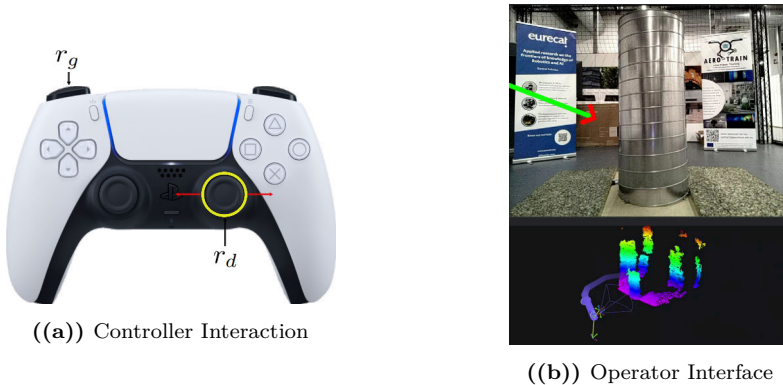


Figure 2.6. Rollercoaster user interaction

The overall interaction paradigm showcasing the architecture with the separate components for drawing the information over the camera stream and moving to the next goal is given in Figure 2.7.

Results also include a user testing phase, during which 35 participants were recruited from the research institution and through word of mouth. The participants' ages ranged between 24 and 61 years old (Mean = 33.61, SD = 9.33). They were requested to rate their UAV piloting experience on a scale of 1 to 5, where 1 indicated no experience and 5 denoted proficiency (Mean = 1.77, SD = 1.19). Each user was asked to perform a navigation test in a highly cluttered simulated environment with and without the Rollercoaster. Then, the users were asked to complete the NASA-TLX questionnaire [60], and their success rate was analyzed (see Figure 2.11), showcasing an average double performance improvement (see Figure 2.8(a)), with up

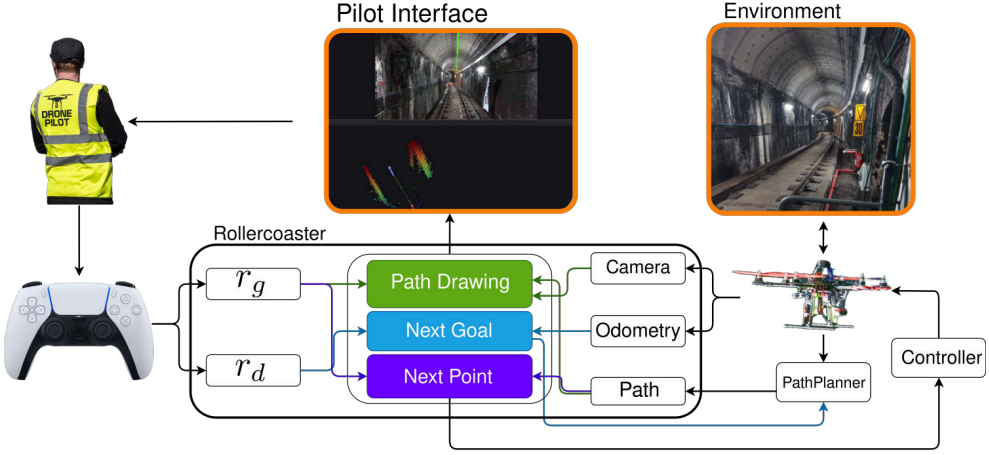


Figure 2.7. Diagram depicting the interaction between the operator and the UAV.

to four times better results when the pilot considered themselves non-experts (see Figure 2.8(b)). The NASA-TLX survey (see Figure 2.9) showcased lower overall demand in navigating with the Rollercoaster compared to manual piloting experience.

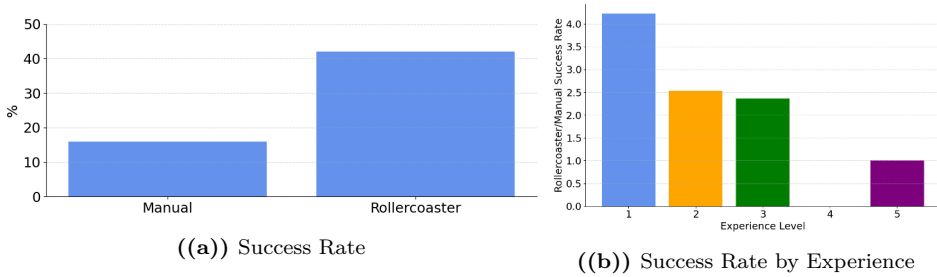


Figure 2.8. User Study Results

Empirical evaluations were also conducted to demonstrate the effectiveness of Rollercoaster with a real UAV. The evaluation was performed using a FPV-like drone equipped with a Voxl2 [61], which integrates PX4 [62], and a RealSense d435 [63] serving as the primary depth sensor. The test setup is illustrated in Figure 2.10, while the operator's perspective is shown in Figure 2.11(a), and the trajectory with the operator's commands is depicted in Figure 2.11(b).

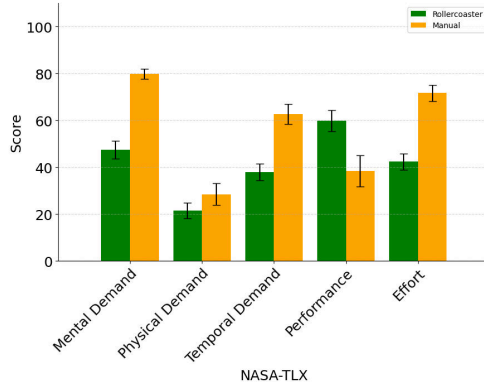


Figure 2.9. Nasa-TLX results. Error Bars are standard error

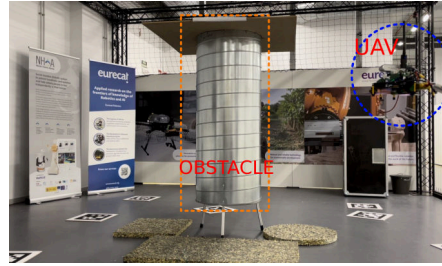


Figure 2.10. Experimental Setting

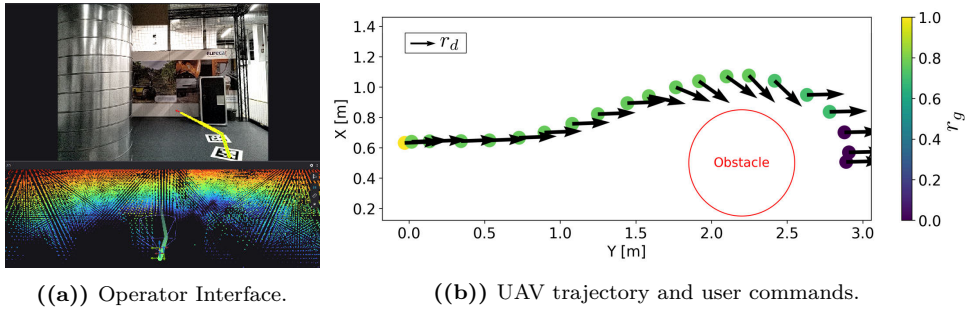


Figure 2.11. Experimental Evaluation

2.4 Point Segment and Inspect

The fourth work aims to simplify the process of visually defining and inspecting various types of infrastructure. Examples of infrastructure that often require visual inspections include bridges, wind turbines, and industrial facilities. During the inspec-

tion mission, it is crucial to ensure coverage of the infrastructure and maintain data consistency, such as capturing images at a consistent distance and orientation. However, achieving this level of precision can be challenging, even for expert pilots. To facilitate UAV piloting and guarantee data integrity, the following work is proposed:

(P4) “*Point, Segment, and Inspect: Leveraging Promptable Segmentation Models for Semi-Autonomous Aerial Inspection*” by Riccardo Franceschini, Javier Rodriguez Marquez, Matteo Fumagalli, and Julian Cayero Becerra, accepted at RO-MAN 2024, Pasadena, California, USA [50]

This method aims to exploit the latest advancement in promptable segmentation models, such as the Segment Anything Model (SAM) [64], to establish a collaborative visual inspection paradigm. In this paradigm, the operator defines the object they want to inspect by specifying positive and negative points over the continuously streamed camera feed. This process provides visual feedback over the camera stream, displaying information about the selected points and the forecasted segmented area. Once a satisfactory segmentation is achieved, the operator can trigger a pipeline to extract the spatial representation of the segmented object from the retrieved depth information. This representation is then used to generate a path over the segmented surface by solving the traveling salesman problem with a 2-opt algorithm [65] (Figure 2.14). The operator then triggers UAV movement to the points, which can be fully autonomous or manually controlled by the operator for translation between points. To orchestrate this interaction, a set of statuses is proposed as shown in (2.1).

$$\begin{aligned} \mathcal{S} = \{ & \textit{Free_Flight}, \\ & \textit{Aim_To_Surface}, \\ & \textit{Plan_To_Surface}, \\ & \textit{Auto_Move_To_Surface}, \\ & \textit{Manual_Move_To_Surface} \} \end{aligned} \quad (2.1)$$

A schema representing the interaction between the UAV and operator through the proposed approach is depicted in Figure 2.12. This diagram illustrates the interaction dynamics between the operator and the UAV. The suggested module receives input points, status indications, or manual waypoint commands issued by the operator, while simultaneously acquiring odometry data and camera information from the UAV. Thus, the proposed layer is responsible for handling perception, segmentation, and path planning while keeping the operator informed through visual cues.

The operator’s control of the UAV is overviewed in Figure 2.13, showcasing the actions needed to control the system, from point definition with the thumbstick to status control with the arrows.

Throughout the operation, the operator is kept informed with information representing the controlled and selected points along the proposed segmentation Figure 2.14(a), the retrieved path Figure 2.14(b), and the completion percentage while moving Figure 2.14(c).

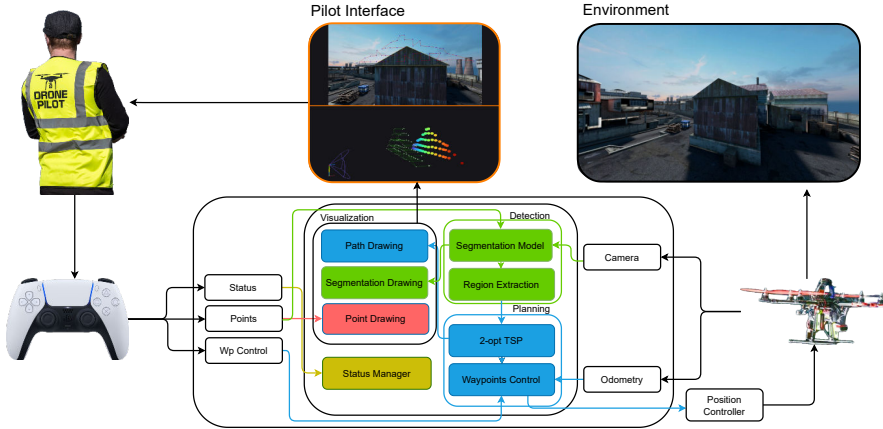


Figure 2.12. Diagram depicting the interaction between the operator and the UAV

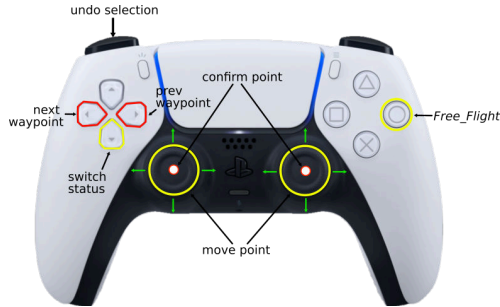


Figure 2.13. Interaction paradigm for Point and Segmentation

To assess the approach, evaluations were carried out in both a photo-realistic environment using Flightmare [66] running as segmentation model SAM-HQ [67] and on a physical platform powered by Voxl2 [61] with FastSAM [68] as segmentation model. Anecdotal evaluations were conducted against a pipe mockup and a wall patch, as depicted in Figure 2.15, demonstrating the system's capability to accurately detect, segment, and inspect the desired object with minimal human intervention in the decision-making process.

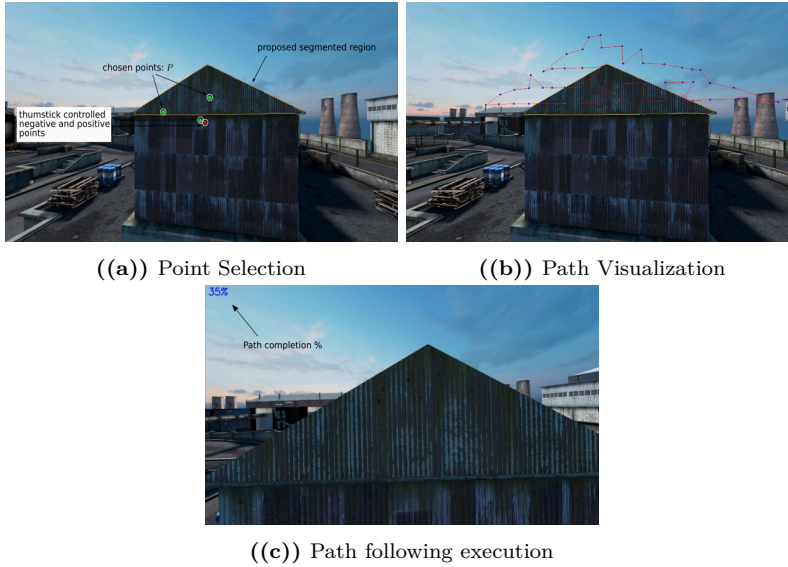


Figure 2.14. Representation of the different interaction steps

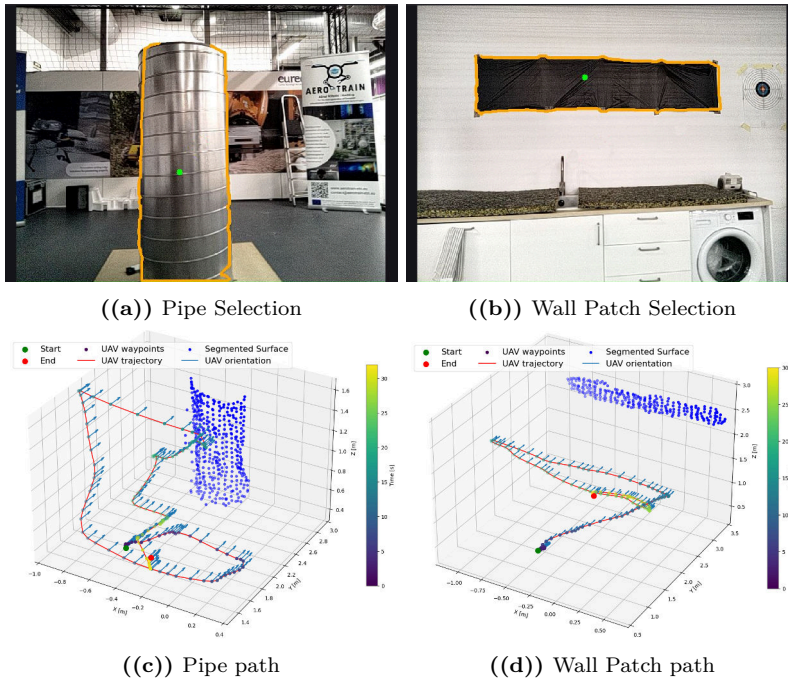


Figure 2.15. Point and Segment experimental evaluation

2.5 AeroAssistant

The latest work introduces AeroAssistant, a framework capable of amalgamating prior research into a teleoperation framework, which offers a familiar teleoperation experience while simultaneously providing advanced functionalities akin to those previously presented. Thus, the AeroAssistant architecture and capabilities are detailed in:

(J1) “*AeroAssistant: A Modern and Flexible Teleoperation Framework*” by Riccardo Franceschini, Matteo Fumagalli, and Julian Cayero Becerra, submitted at *ACM Transaction on Human-Robot Interaction* [69]

The fundamental concept underlying AeroAssistant is to propose a modular, efficient, and flexible architecture for robot teleoperation. Consequently, the framework’s general architecture is illustrated in Figure 2.16, comprising three main modules: the Middleware, responsible for providing platform-agnostic high-level interfaces such as those outlined in [70]; the RemoteControl, tasked with interacting with the operator control device and interpreting user commands into inputs meaningful for the system; and AeroAssistant itself, which interacts with UAV sensors and other processes while offering advanced features, including those previously mentioned, to the operator. A more in-depth explanation of the underlying structure, is provided in Chapter 3. This chapter explains how various features are implemented, the adjustments made for effective human-drone interaction, shares experiences from deploying the AeroAssistant in the field, and outlines current areas of investigation.

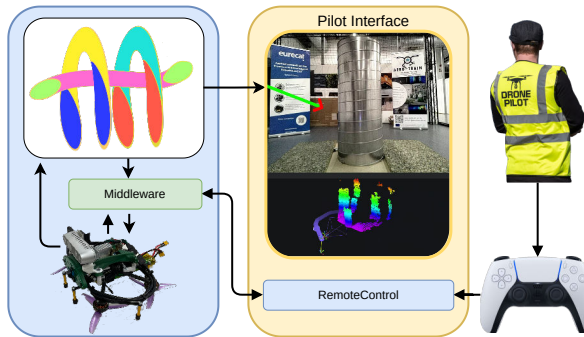


Figure 2.16. AeroAssistant communication architecture

CHAPTER 3

AeroAssistant Core

This chapter explores the AeroAssistant (Figure 3.1), providing a detailed analysis of the proposed framework. It also discusses the deployment experience across different platforms and environmental conditions, as well as current research areas. This expands on the following work:

(J1) “*AeroAssistant: A Modern and Flexible Teleoperation Framework*” by Riccardo Franceschini, Matteo Fumagalli, and Julian Cayero Becerra, submitted at *ACM Transaction on Human-Robot Interaction* [69]

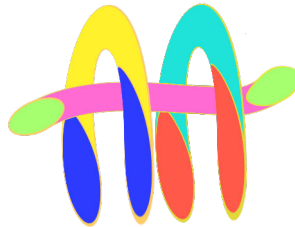


Figure 3.1. AeroAssistant Logo

At first, the RobotStatus (Sec. 3.1), Middleware (Sec. 3.2), and RemoteControl (Sec. 3.3) components are introduced. Subsequently, we examine the framework’s architecture (Sec. 3.4) and plugin definition (Sec. 3.5). We then discuss the adjustment required for effective interaction with the operator during flight (Sec. 3.6), examine modifications made to facilitate operation on low-power devices (Sec. 3.7), and review the interaction proposed during the field test in the final demo of the Aero-Train project (Sec. 3.8). Finally, we conclude by presenting a glimpse of current research with a communication bridge for MR interfaces (Sec. 3.9), aimed at future immersive interfaces. The framework underwent testing on various hardware platforms, including LattePanda Delta3 [71], Intel Nuc [72], and Voxl2 [61]. Despite performance variations across these platforms, AeroAssistant consistently managed all computations, maintaining a satisfactory framerate of 30Hz on the operator interface. The proposed framework requires a version of C++ 17 or higher and relies solely on Open3D [73] for spatial data manipulation and OpenCV [74] for image manipulation. For AeroAssistant to function, it is assumed that the UAV has the ability

to perceive its three-dimensional environment, possesses an RGB camera, and can estimate its position and orientation in space as $\mathbf{p}_r \in \mathbb{R}^3$ and $\mathbf{q}_r \in \mathbb{H}$.

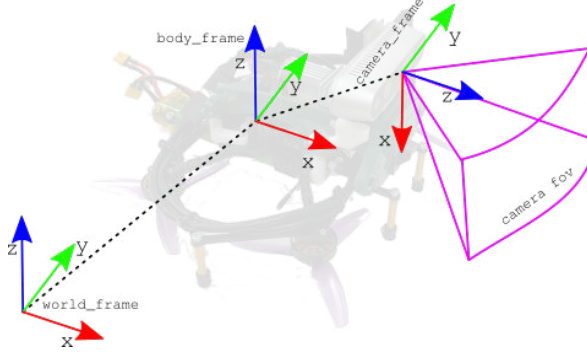


Figure 3.2. A schematic representation of the UAV frame configuration

3.1 Robot Status

To ensure seamless coordination within the system and reflect these changes in both the UAV behavior and the remote pilot interface, it is essential that all components remain informed about the current status. Therefore, we propose the implementation of a state machine capable of encoding the various statuses of the drone. The robot's status s_r consists of a set of base states \mathcal{S} , which are independent of any plugin implementation, each representing a specific status:

$$\begin{aligned} \mathcal{S} = \{ & \text{Idle}, \\ & \text{TakeOff}, \\ & \text{Land}, \\ & \text{Free_flight} \} \end{aligned} \tag{3.1}$$

These statuses enable pilots to maintain basic control over the UAV, similar to a standard teleoperation mechanism, encompassing functionalities such as autonomous landing and takeoff. To manage synchronization among the various actors of the framework, we employ a request-reply architecture, with the Middleware acting as an intermediary between operator control and AeroAssistant. This setup ensures efficient message passing and confirmation delivery, as depicted in Figure 3.3, thereby ensuring correct synchronization among all actors across different statuses and allowing bidirectional changes in the framework status from both AeroAssistant and RemoteControl. This architecture grants operators the flexibility to consistently modify states, thereby maintaining full control of the system. Simultaneously, it empowers the AeroAssistant to execute autonomous behaviors that may involve a sequence

of internally coordinated states. These statuses are promptly communicated to the operator and visualized in the interface, ensuring transparency and awareness.

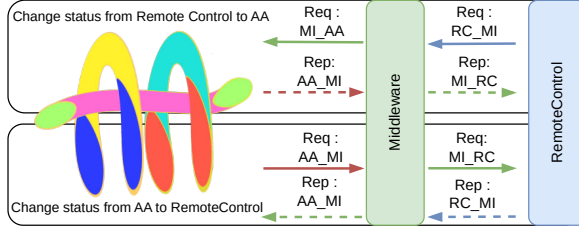


Figure 3.3. Status change communication architecture

3.2 Middleware

The interaction Middleware, situated within the UAV, plays a central role in maintaining communication among all involved entities, including remote pilot input, AeroAssistant, and the UAV itself. Its primary task is to establish an abstraction layer, akin to the one proposed in [70], which exposes common topics tailored to the platform’s specifics, such as specialized routines for arming or issuing commands to the platform. This layer serves as a crucial facilitator for synchronizing communication among all actors, as elucidated in Section 3.1. Through this synchronization, the Middleware enables both RemoteControl and AeroAssistant to publish and subscribe to high-level topics for sending waypoints, arming, or initiating specific routines, while seamlessly managing lower-level adjustments specific to the platform’s requirements under the hood.

3.3 Remote Control

We consider now the RemoteControl feature of the framework, which is essential for delivering a seamless and intuitive navigation experience while providing instant access to the framework’s capabilities. Utilizing its widespread availability and ease of customization, we have chosen to utilize the SONY DualSense™ Wireless Controller [59] as our reference point. However, integrating other RC controllers is equally feasible without any complications.

To operate, the remote controller must be connected to a companion computer, responsible for managing all message passing with the Middleware. Usually, the computer and the operator’s receiving device are the same, although this is not a strict requirement. Given AeroAssistant’s aim to provide a flexible framework that offers an enhanced yet familiar user experience, the behavior of the controller needs

to adapt depending on the state s_r . Thus, by default, the operator is presented with the following configuration, as illustrated in Figure 3.4.

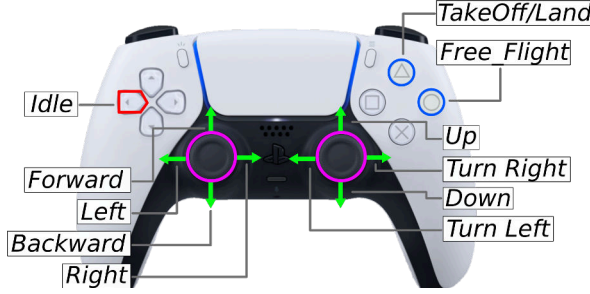


Figure 3.4. Standard interaction paradigm

In this setup, the operator can take control and arm the UAV using the left arrow button, which sets the UAV status to $s_r = Idle$. Pressing the triangle button transitions the status to $s_r = TakeOff$, initiating an autonomous takeoff operation at a predefined height h m. Upon pressing the triangle button again, the status shifts to $s_r = Land$, prompting autonomous landing by descending. To initiate standard teleoperation in position control, the operator presses the circle button, setting the status to $s_r = Free_Flight$.

3.4 Manager Architectures

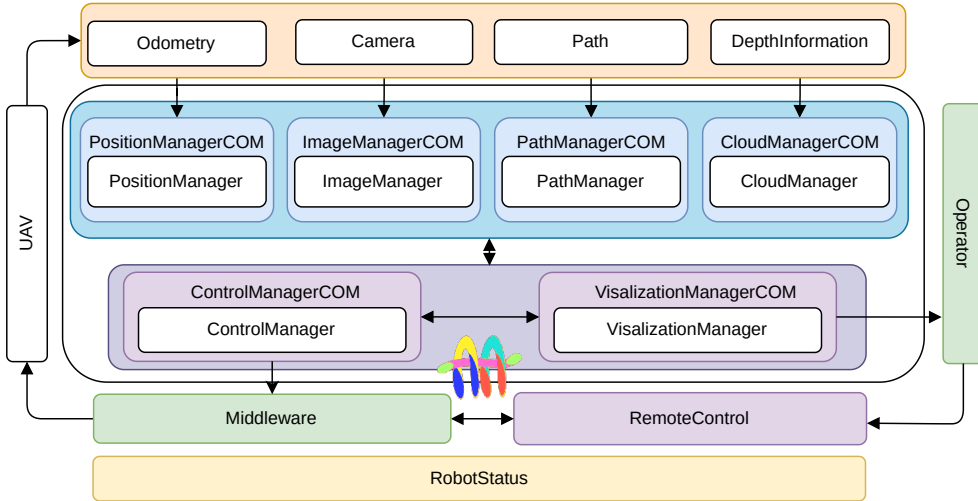


Figure 3.5. AeroAssistant Architecture

This section provides a thorough examination of the framework architecture shown in Figure 3.5. The AeroAssistant comprises a series of specialized managers, each dedicated to a specific task. The communication between these managers and other components outside of AeroAssistant is abstracted through a series of communication managers. This abstraction ensures potential interoperability with various communication paradigms such as ROS1 [75], ROS2 [76], and ZMQ [77], while maintaining the underlying implementation agnostic and thus interoperable.

The managers can be categorized in two macro categories, the passive managers and the active managers. The passive managers (Sec. 3.4.1) are tasked with efficiently retrieving and processing incoming information from the UAV. They then expose meaningful data to the active managers (Sec. 3.4.2). Conversely, the active managers are responsible for sending actual commands to the UAV and keeping the operator informed during operation.

3.4.1 Passive Managers

The passive manager encompasses all activities that do not involve interaction with either the UAV or the operator. Therefore, in this category, managers such as PositionManager, ImageManager, PathManager, and CloudManager are included, as highlighted in Figure 3.6.

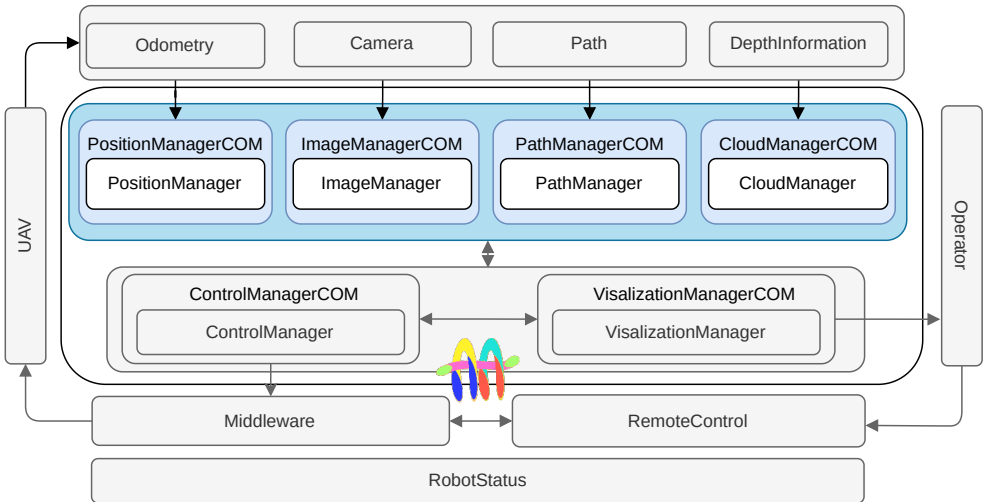


Figure 3.6. Passive Managers

These managers serve as crucial intermediaries between the processes and sensors operating within the UAV, playing a central role in processing incoming information and providing accessible methods for other managers and processes to effectively

utilize the processed data. This architectural design offers significant advantages, particularly in reducing overhead and enabling parallel computation, which is especially valuable when onboard computation resources are limited. Ultimately, this aims to deliver a seamless experience for the operator.

Internally, these managers process their data using a queue system with a maximum capacity of n elements, coordinated with a mutex system to handle concurrent resource allocation and ensure synchronization between data retrieval and processing. A schematic depiction of a generic passive manager is provided in Figure 3.7, illustrating the data flow. Initially, raw UAV information is converted into a generic format independent of the communication paradigm used. Subsequently, the data undergoes a filtering process to render it usable by the rest of the system. The filtered data is then stored in the CleanData queue, while the RawData queue continues to receive updates. It's crucial to note that these processes, including data reception and filtering, occur in parallel, with resource coordination facilitated by mutexes.

Depending on the specific manager, various types of data may need extraction, possibly combining information from other managers. Consequently, internally, managers implement necessary functions to extract data, which could range from estimating the difference in orientation to the closest point based on current position and spatial information to generating a traversal plan over a segmented point cloud. Once extracted, the information is stored and exposed through a series of GetData methods, ensuring proper concurrency management through mutexes.

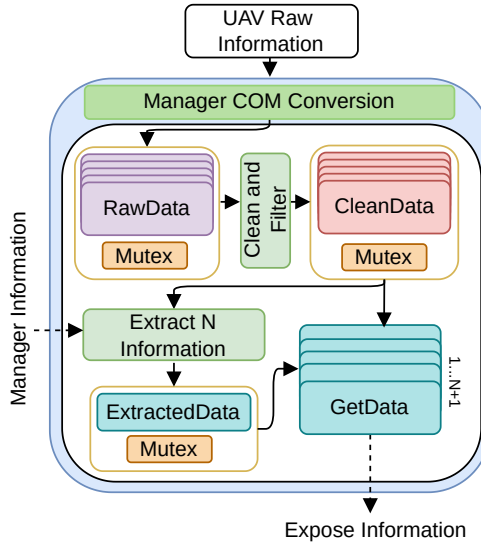


Figure 3.7. General Architecture of a Passive Manager

3.4.2 Active Managers

In the teleoperation framework, in addition to the passive managers, another essential part includes the active managers. Specifically, these are the ControlManager and the VisualizationManager, as shown in Figure 3.8. These managers play dynamic roles in the system, each serving distinct yet complementary functions.

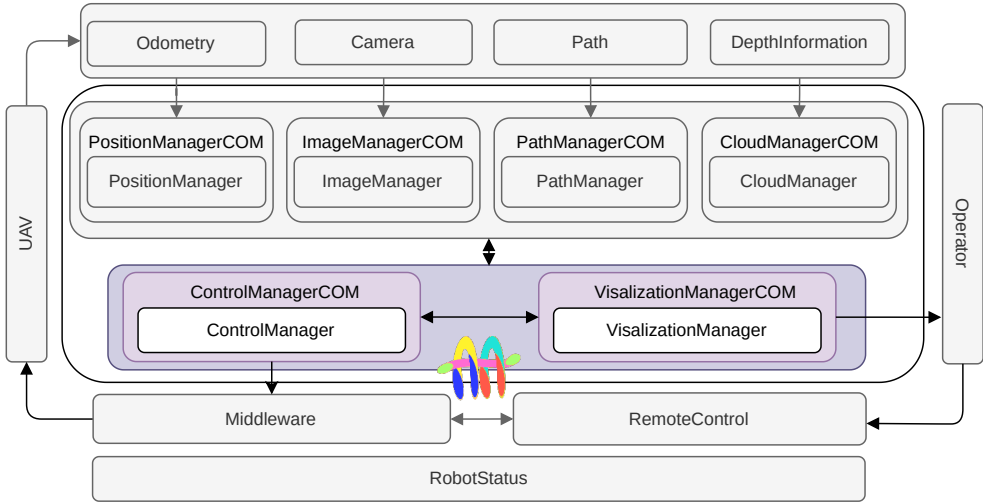


Figure 3.8. Active Managers

The ControlManager assumes a pivotal role in the teleoperation setup, acting as the channel through which commands are conveyed to the UAV. It serves as the interface between the operator and the UAV, facilitating real-time control and maneuvering. By interfacing with the Middleware layer, the ControlManager ensures seamless communication between the operator's commands and the UAV's actions, effectively translating high-level directives into actionable instructions for the vehicle's systems. Conversely, the VisualizationManager assumes the responsibility of maintaining the operator's awareness regarding the ongoing processes within the UAV. Serving as the visual feedback mechanism, it provides real-time updates and status reports to the operator. Through intuitive graphical representations, the VisualizationManager enhances the operator's situational awareness regarding the current UAV status and active processes. This allows operators to make informed decisions and adjustments as necessary, empowering them to effectively monitor and supervise the UAV's operations. The ControlManager and the VisualizationManager together form the active managerial component of the teleoperation framework, serving as the bridge between human operators and unmanned aerial vehicles. They work together to facilitate efficient and effective teleoperation, empowering operators to gain precise control over UAVs while maintaining a comprehensive understanding of their operational environment. Therefore, these Managers receive information from both the

other passive managers and the operator, translating this information into actions for both the UAV and the operator interface. Consequently, they do not directly interface with sensor data but instead collect information from all other involved actors. Their behavior is directly linked to the status s_r . Thus, a generic scheme of the active managers is proposed in Figure 3.9. These two processes of visualization and control are executed separately and coordinated according to s_r , as described in Section 3.1. The ControlManager retrieves the current UAV position and any necessary spatial information from the passive manager. This information may vary from simple orientation offsets to a series of waypoints. It is then combined with s_r and user input to determine the desired UAV position, which is converted to the correct communication format and sent to the Middleware. Meanwhile, the VisualizationManager retrieves the CameraImage and DataToVisualize from the PassiveManagers. The information to visualize can range from simple text or geometric structures to more complex 3D point clouds that need to be projected onto the 2D plane. To achieve this, the CameraInfo along with s_r are used to craft the enhanced camera feedback, which is converted to the correct communication format and ultimately sent to the operator.

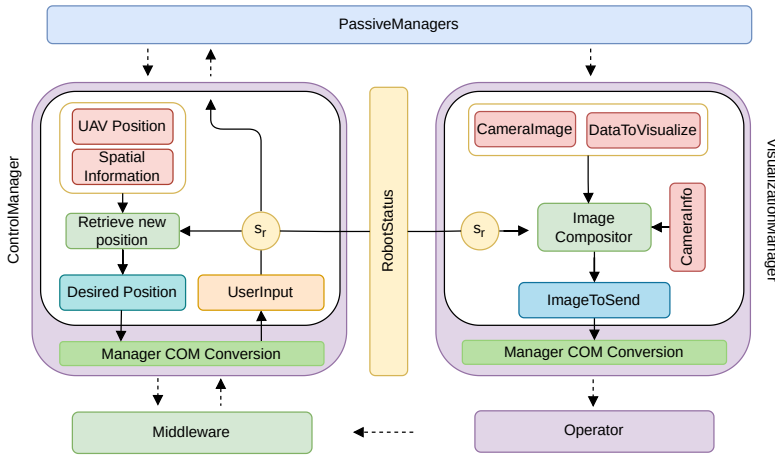


Figure 3.9. General Architecture of the Active Managers

3.5 Plugins

This section aims to expose the functionalities currently being deployed within the AeroAssistant, showcasing the interaction paradigms used both from a visualization point of view and from a remote control perspective. It also describes how S is expanded to include multiple functionalities and how those additional states are used to adjust visualization and remote control behaviors. Therefore, the following sec-

tion addresses some of the plugins implemented, some of which have already been explained and evaluated in separate publications. For this reason, the evaluation now focuses more on the integration with the framework rather than the functionality and performance itself.

3.5.1 Lock to a point

A common action required when piloting a UAV is the ability to select a point on the camera feedback, lock onto that point, and maintain orientation and position towards it, counteracting possible degradation in position estimation, akin to visual servoing. Therefore, this feature is integrated within the framework. To manage the different states, first, the robot's states are updated as follows:

$$\mathcal{S} = \{\mathcal{S} \cup \{Choose_Target, Following_Target\}\} \quad (3.2)$$

The alternation between $s_r = Choose_Target$ and $s_r = Following_Target$ is carried out by the operator with the controller schema depicted in Figure 3.10. Pressing the up arrow changes the status to $s_r = Choose_Target$. After the arrow up is pressed



Figure 3.10. Controller interaction for tracking a point

for the first time, the operator defines the area to track by maneuvering an overlaid window of dimensions $n \times n$ pixels using the right thumbstick over the camera stream, as demonstrated in Figure 3.10. Once the area of interest is delineated, the operator initiates a Kernel Correlation Filter [78] (KCF) by pressing the arrow up on the controller, covering the previously defined patch. This filter facilitates tracking of any image patch across the camera stream, provided there are sufficient features to track between consecutive frames. Throughout the operation, the operator receives visual cues overlaid on the camera stream, as illustrated in Figure 3.11, featuring varying visualizations contingent upon the success or failure of the tracking process.

The center of the tracked patch is then utilized to extract the position $\mathbf{p}_i \in \mathbb{R}^3$ and normal $\mathbf{n}_i = \{n_{ix}, n_{iy}, n_{iz}\}$ of the point of interest, relying on the depth information retrieved from the previously described CloudManager. Subsequently, the point and normal estimations are averaged over k data samples to mitigate potential drift in

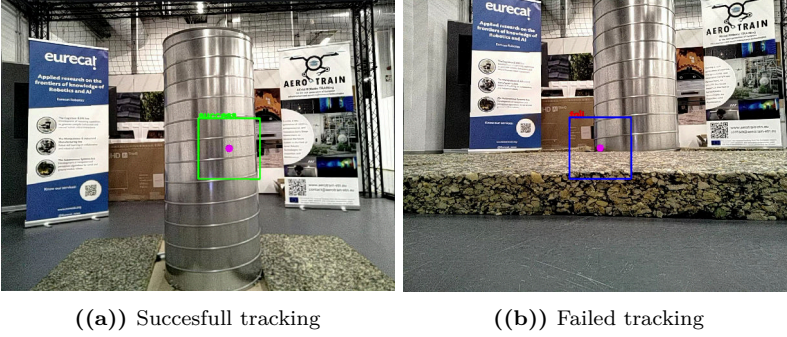


Figure 3.11. Different tracking visualizations

position and orientation, yielding the averaged values $\bar{\mathbf{p}}_i$ and $\bar{\mathbf{n}}_i$. These averaged values are then utilized to determine the desired position and orientation of the UAV, computed as a position at a distance γ m from the point of interest.

$$\mathbf{p}_d = \bar{\mathbf{p}}_i + \gamma \bar{\mathbf{n}}_i \quad (3.3)$$

$$\mathbf{o}_d = -\bar{\mathbf{n}}_i \quad (3.4)$$

The extracted values are then used to maintain the position \mathbf{p}_r and orientation \mathbf{q}_r of the UAV in front of \mathbf{p}_i . During the all process the operator is also aware trough a 3D panel showcasing the extracted \mathbf{p}_d as in Figure 3.12.

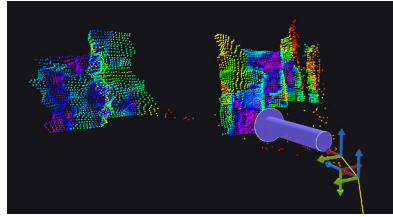


Figure 3.12. 3D panel visualization

Furthermore, interaction can be enhanced by integrating an autonomous detection mechanism responsible for autonomously retrieving \mathbf{p}_i . This leaves the operator only responsible for initializing detection and ensuring the correctness of the entire approach. More details on autonomous detection are provided in Sec. 3.8. A schematic representation of the state transition is shown in Figure 3.13.

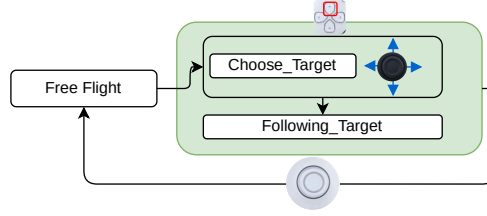


Figure 3.13. Transition between different states S for tracking a point

3.5.2 Move to a point

Another relevant plugin developed adds the feature of selecting a point in the image plane and guide the aircraft towards it, ending up at a position that is at a user-defined distance γ m from the obstacle and oriented towards the point of interest. In this case, the robot's state is updated as follows:

$$\mathcal{S} = \{\mathcal{S} \cup \{Aiming_To_Target, Moving_To_Target\}\} \quad (3.5)$$

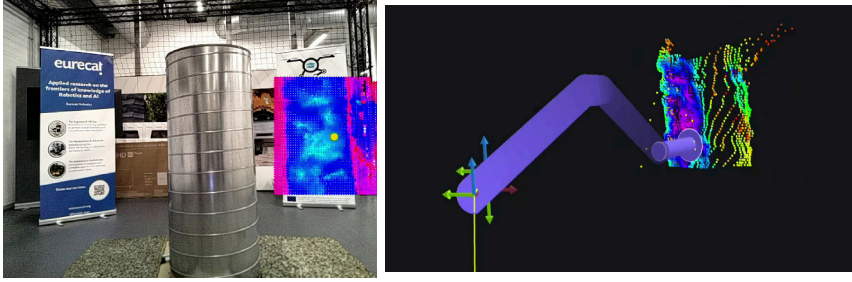
The selection and the change of status to $s_r = Aiming_To_Target$ are triggered by pressing the square button on the controller. In this visualization, the operator has to define a point $\mathbf{p}_m \in \mathbb{R}^2$ on the image plane.



Figure 3.14. Controller interaction for moving to a point

As shown in Figure 3.14, the operator is presented with a visualization that encodes information about the normal orientation of the surface within the proximity of \mathbf{p}_m using a rainbow color-scheme, with \mathbf{p}_m itself represented as a yellow dot, as in Figure 3.15(a). This provides immediate feedback on the geometric structure of the selected area and the feasibility of the selected point. Then, the point \mathbf{p}_d and orientation \mathbf{o}_d are extracted as in (3.3) and (3.4).

Given the UAV's need to navigate autonomously in space, a planner and mapping approach capable of retrieving collision-free paths are essential. Consequently, the same methods previously described [57] and [58] used in the the Rollercoaster (Sec. 2.3) are adopted. The retrieved path is visualized in the 3D panel as shown in



((a)) Camera Visualization when aiming to a point p_m ((b)) 3D visualization with the extracted region, retrieved path and p_m

Figure 3.15. Different move to a point visualizations

Figure 3.15(b). Upon pressing the square button again, the state transitions to $s_r = \text{Moving_To_Target}$, initiating the path following procedure. Consequently, the Path-Manager provides the path, defined as a set of N points $\mathcal{P} = \{p_r, p_0, p_1, \dots, p_d\} \in \mathbb{R}^3$, to the ControlManager. The ControlManager then guides the UAV by transmitting the next waypoint p_{k+1} only when the UAV is within r m from p_k . A schematic representation of the transition between states is given in Figure 3.16

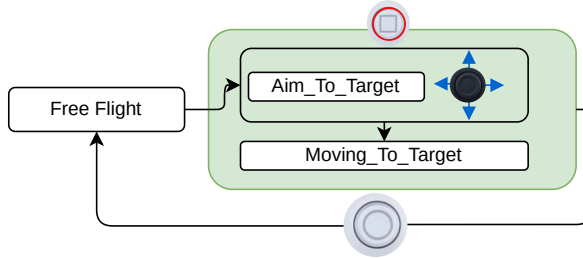


Figure 3.16. Move to a point state S transitions.

3.5.3 Align and Follow Surface

Another relevant feature deployed within AeroAssistant is the ability to align against any surface maintaining distance and attitude with respect to it, while also being able to perform vertical and lateral displacements. Detailed descriptions of this plugin are provided in previous work [54] and briefly summarized in Sec. 2.2. In this section, the focus will be on the integration with the system and adaptation to fully integrate it with respect to the original work. Thus, to include this behavior within the system, the set of possible statuses is updated as follows:

$$\mathcal{S} = \{\mathcal{S} \cup \{\text{Align}, \text{Lock_To_Surface}\}\} \quad (3.6)$$

The interaction with the remote controller is depicted in Figure 3.17. Unlike the previous implementation where the operator triggered functions with a button on a table-like interface, in this case, the operator triggers the following functions with a controller. We argue that this implementation contributes to a more natural experience and interaction with the system compared to the previously published interaction as the operator does not have to switch to another control paradigm. To align with the closest point \mathbf{p}_c the UAV position \mathbf{p}_r , the operator simply presses the cross symbol on the controller, setting $s_r = \text{Align}$. This computes the yaw offset η° to align with the closest point \mathbf{p}_c and sends the updated orientation to the position controller. During operation, the camera feedback highlights the closest point \mathbf{p}_c as a circle on the camera stream, with colors changing to reflect the angle offset η° (green if $|\eta^\circ| < 5^\circ$, yellow between $5^\circ \leq |\eta^\circ| \leq 15^\circ$, and red otherwise).



Figure 3.17. Control interaction for align and translate

To change the status to $s_r = \text{Lock_To_Surface}$, the operator initiates and maintains the status by pressing the right trigger on the controller. This automatically stores the current distance $d_{\mathbf{p}_c}^t$ to the closest point \mathbf{p}_c at time t , and aligns the UAV to the closest point as described earlier. Unlike in previous work, where translation was binary (either lateral or vertical), the operator now has more fine-grained control, with l_c, v_c both in the range $[-1, 1]$ representing lateral and vertical displacement inputs. This allows for precise and controlled velocity adjustments of the UAV's movement. The current position $\mathbf{p}_r^t = [p_x^t, p_y^t, p_z^t]$ at time $t + 1$ is updated as:

$$p_y^{t+1} = p_y^t + \delta l_c \quad (3.7)$$

$$p_z^{t+1} = p_z^t + \delta v_c \quad (3.8)$$

with δm representing the maximum translation. Then, given the new position \mathbf{p}^{t+1} , a new closest point \mathbf{p}_c^{t+1} , its distance $d_{\mathbf{p}_c}^{t+1}$, and normal \mathbf{n}_c^{t+1} are estimated using the current point cloud. The alignment angle η° is retrieved and used to adjust to the new position while the distance is corrected as follows:

$$p_x^{t+1} = p_x^t + (d_{\mathbf{p}_c}^t - d_{\mathbf{p}_c}^{t+1})n_x^{t+1} \quad (3.9)$$

The updated position \mathbf{p}_r^{t+1} and orientation η° are sent to the position controller. Throughout this process, the operator is informed of the current alignment via a

heatmap visualization of the UAV's attitude relative to the facing surface, in addition to the previously described dot representing the closest point. A schematic representation of the transition between states is given in Figure 3.18.

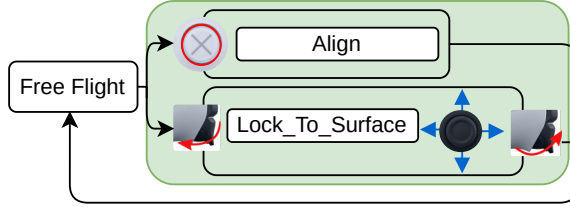


Figure 3.18. Align and Lock to Surface state transitions.

3.5.4 Rollercoaster

Piloting an aircraft is inherently challenging, especially when navigating in cluttered environments with limited situational awareness. To address this issue, the Rollercoaster [56] previously described in Sec. 2.3 is integrated within the system as a plugin. In this method, the pilot controls the direction and velocity of the UAV while a planner in the background is responsible for retrieving collision-free paths and navigating towards a goal. This proposed interaction paradigm has been named Rollercoaster, and accordingly, the set of statuses is updated as follows:

$$\mathcal{S} = \{\mathcal{S} \cup \{Rollercoaster\}\} \quad (3.10)$$

To trigger $s_r = Rollercoaster$, the operator simply uses the left trigger, which is also used to control the velocity. Releasing it will re-establish $s_r = Free_Flight$. As previously described in Sec. 3.5.2, the same planner [57] along with the mapping presented in [58] are used due to their capabilities of reliably and efficiently retrieving a safe path. During navigation, the operator is kept informed with an enhanced visualization (Figure 3.19), which displays the retrieved path (yellow line) over the camera feedback. The current r_g is shown as a green line over the path. Additionally, as described in Sec. 2.3, a spatial representation of the UAV, along with the reconstructed map and retrieved path, is provided (Figure 2.6(b)).



Figure 3.19. Interaction paradigm Rollercoaster

A schematic representation of the transition between states is given in Figure 3.20

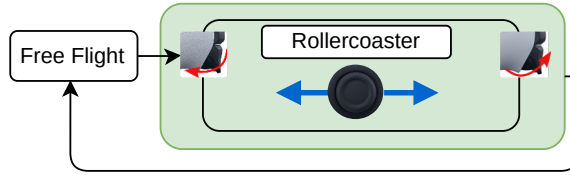


Figure 3.20. Align and Lock to Surface state S transitions.

3.5.5 Point to Segment to Plan

Performing visual inspections of object surfaces can be inherently complex, especially when ensuring coverage and alignment with the surfaces for accurate data collection. Therefore, another plugin developed, studied in detail in [79] and in Sec. 2.4, focuses on simplifying this task. This plugin aims to leverage the latest advancements in promptable segmentation models, such as [64, 67, 68], to streamline the process of defining visible surfaces or objects that require inspection.

Initially, the operator defines an object for inspection by specifying points over the camera image that indicate areas to include or exclude from segmentation, performed by the segmentation model. The model then proposes the segmented area to the operator, who can confirm or refine the segmentation further. Once a satisfactory segmentation is achieved, the spatial geometry of the object is retrieved and used to extract a series of waypoints covering the entire surface. These waypoints are ordered from the PathManager solving the travel salesman problem using the 2-opt [65] algorithm to obtain the shortest path. Given the complexity of the interaction

paradigm, multiple statuses are added:

$$\mathcal{S} = \{\mathcal{S} \cup \{Aim_To_Surface, Plan_To_Surface, Auto_Move_To_Surface, Manual_Move_To_Surface\}\} \quad (3.11)$$

These statuses are orchestrated with the controller interaction scheme represented in Figure 3.21.

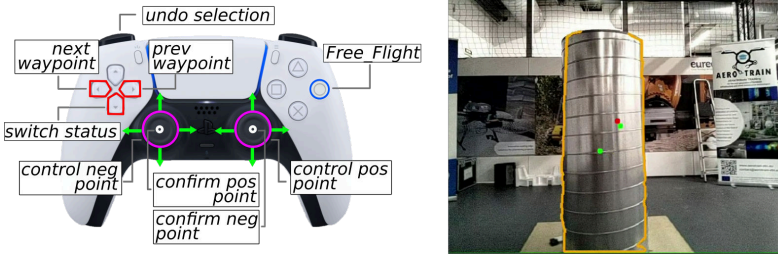


Figure 3.21. Control interaction for Segment and Plan

Pressing the arrow down switches the state to $s_r = Aim_To_Surface$, allowing the operator to define the points used for prompting the model through the thumbstick. Pressing the arrow down again changes the status to $s_r = Plan_To_Surface$, where a spatial representation of the segmented surface is extracted, and a traversal plan at a distance k m facing the segmented surface is proposed to the operator.

To initiate movement, the operator presses the arrow down again, transitioning the status to $s_r = Auto_Move_To_Surface$. Subsequently, the UAV autonomously follows the list of waypoints as described in Sec. 3.5.2. Since the operator may want to retain control or spend more time at a single waypoint or inspect a certain spot again, pressing the arrow down button again changes the state to $s_r = Manual_Move_To_Surface$. At this point, pressing the left and right arrows moves the UAV forward or backward along the list of waypoints, while pressing the circle button returns to $s_r = Free_Flight$. A clearer representation of the status changes is provided in Figure 3.22.

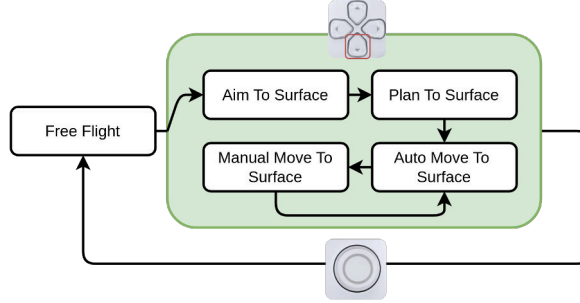


Figure 3.22. This scheme represents the transition between different states (S), with the downward arrow button responsible for switching between states and a circle used to regain control

3.6 Augmented Visualization

Interacting with the UAV through camera feedback poses challenges, especially when dealing with a constantly moving camera feed. Without gimbal or camera stabilization, this dynamic environment can significantly impact the operator's experience. Implementing AR solutions to maintain spatially coherent elements in the operator's feedback is essential to alleviate these problems. However, existing XR frameworks, such as [80–82], are not ideally suited for this scenario. These frameworks are predominantly designed for portable devices such as smartphones, tablets, or HMDs, encompassing their entire pipeline from state estimation to visualization.

Therefore, motivated by the desire to minimize dependencies within AeroAssistant, some fundamental features essential for AR visualization are implemented. These includes accurately representing spatial information within the camera plane and the ability to maintain spatial anchors consistently across consecutive frames, even under camera displacement. These features ensure the seamless integration of objects such as path or point cloud representation within the AR environment, enhancing the overall user experience.

3.6.1 Camera Model

The primary objective of an AR system is to accurately represent an object's position in virtual space relative to the image plane of the camera used to perceive the environment. Achieving this necessitates a thorough understanding of the camera's functionality and characteristics. Accordingly, the camera is typically described using the Pinhole camera model [83,84], depicted at a high-level in Figure 3.23.

This model consider that light passing through a pinhole projects an inverted image onto the image plane. A point $\mathbf{P} = \{x_1, x_2, x_3\} \in \mathbb{R}^3$ is then related to its

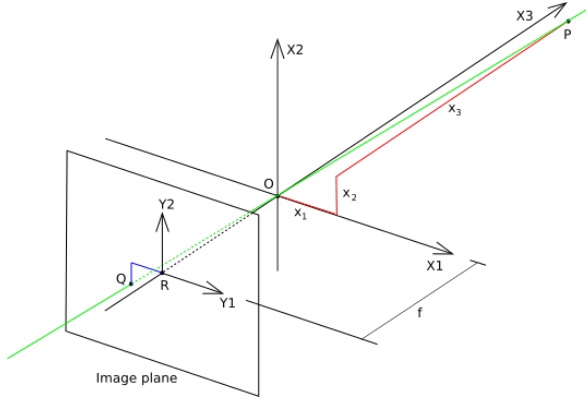


Figure 3.23. Pinhole Camera Model

equivalent on the image plane, $\mathbf{O} = \{y_1, y_2\} \in \mathbb{R}^2$, through the focal distance f as follows:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \frac{f}{x_3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad (3.12)$$

Here, f represents the distance from the origin to the camera plane. However, the model described is applicable to analog cameras; digital cameras differ slightly, requiring the introduction of additional parameters for accurate mapping. While in analog images, the image plane coordinates have their origin at the image center where the x_3 axis intersects the image plane, digital image coordinates usually originate from the upper-left corner of the image. Thus, the parameters c_x and c_y account for the potential translation between image plane and digital image coordinates. Hence, 2D points in the image plane and those in the image are offset by a translation vector $[c_x, c_y]^T$:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \frac{f}{x_3} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix} \quad (3.13)$$

Furthermore, it's essential to consider that images are typically expressed in pixels, necessitating adjustment via a scaling factor k and l (representing units conversion along the two directions of the image plane, when $k = l$, the camera is said to have square pixels). The updated equation is:

$$\mathbf{O} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \frac{f}{x_3} \begin{pmatrix} kx_1 \\ lx_2 \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix} = \begin{bmatrix} fk \frac{x_1}{x_3} + c_x \\ fl \frac{x_2}{x_3} + c_y \end{bmatrix} = \begin{bmatrix} \alpha \frac{x_1}{x_3} + c_x \\ \beta \frac{x_2}{x_3} + c_y \end{bmatrix} \quad (3.14)$$

However, for convenience, the camera characteristics are often represented using a matrix formulation that encodes the information into the camera matrix. This matrix describes the camera model and operates on homogeneous coordinates:

$$\begin{pmatrix} y_1 \\ y_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} \alpha & 0 & c_x & 0 \\ 0 & \beta & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \quad (3.15)$$

This matrix can be further decomposed as:

$$\begin{pmatrix} y_1 \\ y_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} \alpha & 0 & c_x \\ 0 & \beta & c_y \\ 0 & 0 & 1 \end{bmatrix} (\mathbf{I} \ 0) \mathbf{P} = \mathbf{K} (\mathbf{I} \ 0) \mathbf{P} \quad (3.16)$$

Here, \mathbf{K} is commonly referred to as the camera matrix. This model could be further expanded to consider lens distortion or camera skewness; however, for conciseness, these factors are not included.

3.6.2 From Virtual to Real

To enhance the operator experience and facilitate interaction with the UAV, the AeroAssistant is tasked with translating spatial information retrieved by the UAV into the operator's camera stream. As depicted in Figure 3.24, the AeroAssistant receives information from passive managers in the world frame that requires translation into the camera frame to be correctly represented. This translation is known from the odometry information, which tracks the UAV's current position relative to the initial global frame. Subsequently, the translated data is projected onto the raw camera images using camera information, and finally, it is transmitted to the communication manager for conversion.

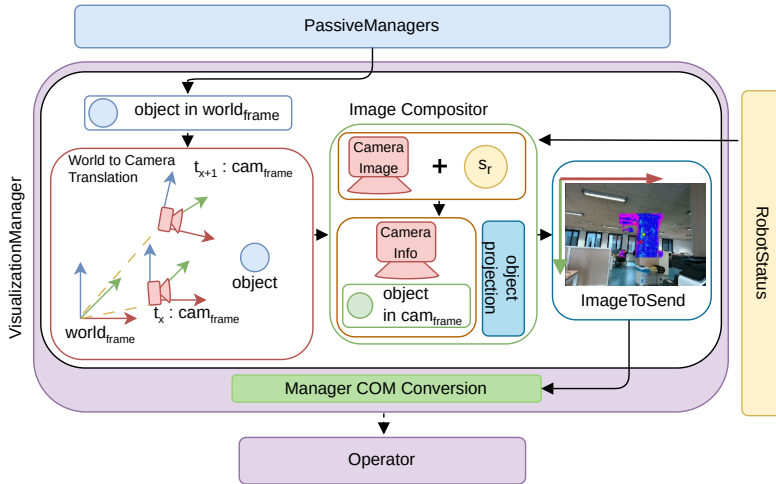


Figure 3.24. Mapping from 3D to camera

Such approach of maintaining the processed data in world frame and then translating to camera frame at visualization time is necessary to allow correct representation of the data with respect to the current point of view as in Figure 3.25 where the retrieved inspection path is correctly represented over the segmented door independently from the current UAV position.

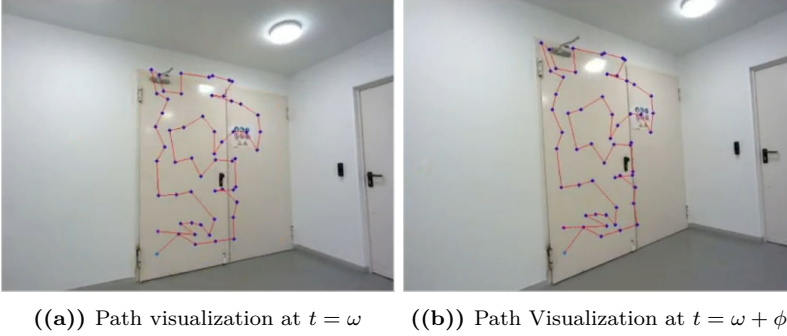


Figure 3.25. Path visualization under different camera position

3.6.3 Interacting While Flying

Ensuring accurate data translation is not only imperative for visualization during operator interaction but also fundamental when activating the segmentation model during flight, as elaborated in Sec. 3.5.5. Throughout the flight, the UAV experiences inherent oscillations even when attempting to maintain a stable position. Without the compensatory measures outlined in Sec. 3.6.2, this oscillation poses two interconnected challenges. Firstly, as previously discussed, incorrect perception arises for the operator regarding the prompt made. Failure to update the point's position results in a disparity between its perceived location in the camera plane and its actual position. Secondly, the segmentation network relies heavily on the accuracy of these prompts within the 2D image. Inaccuracies in updating these prompts lead to erroneous queries. Consequently, at every frame, the list of queried points undergoes reprojection and updating to align with the camera frame and subsequently relayed to the segmentation model. The final result is a point which is correctly updated and represented in the operator camera feedback (Figure 3.26) and such change in position is reflected also in the prompted points. Since the AeroAssistant is taught to run on any device the inference of the segmentation model is considered external to the all framework while the two parts are connected through the communication managers agreeing on a specific message structure exchange.

This approach is particularly suitable, as deep learning models typically rely heavily on the hardware integrated into the UAV. By adopting this method, we can achieve a flexible architecture capable of adapting to various hardware configurations. A schematic representation of the communication flow among all involved actors is

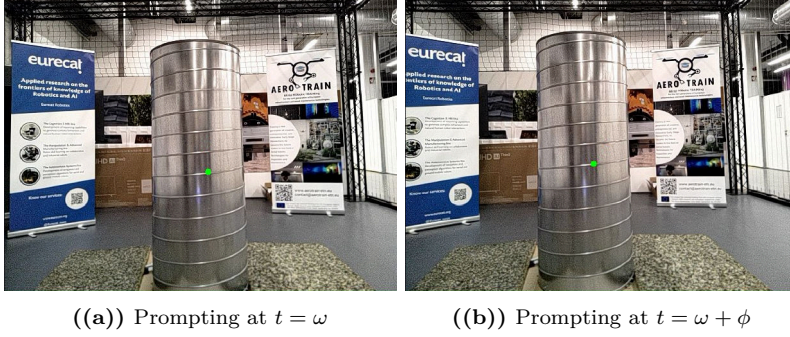


Figure 3.26. Prompting at different moment in time

provided in Figure 3.27, illustrating the data flow for maintaining the consistency of the prompted points across different consecutive frames. Thus, the CloudManager is responsible for maintaining a list of prompted points in the world frame along with their labels. It also utilizes the odometry and camera information from the other passive manager to continuously translate the points, first to the camera frame and then to project them onto the camera plane before sending them to the visual segmentation node.

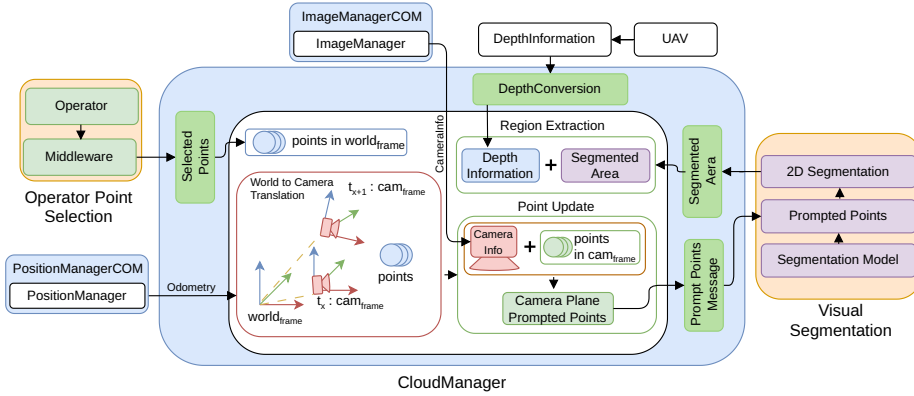


Figure 3.27. Segmentation Points update schema

3.7 Running on Constrained devices

The dimensions of a UAV can vary significantly, from lightweight multirotors weighing just a few grams to huge helicopters or fixed-wing platforms capable of carrying several kilograms of payload. However, as flying platforms with limited power capa-

bilities, the ability to carry out necessary computations on smaller devices can offer unique advantages. Therefore, finding the optimal balance between weight, power consumption, and capacity is crucial for extending flight time and ultimately enhancing the range of tasks that can be accomplished. Thus, within the domain of computing boards commonly used in the aerial platform, examples include the Nvidia Jetson Orin [85], Raspberry PI4 [86], Intel Nuc [72], and LattePanda Delta3 [71], among others. Each of those boards has its advantage or disadvantage which could range from capabilities of running easily advanced deep learning algorithm for the Nvidia Jetson Orin thanks to its Cuda [87] compatibility, to compatibility with any existing software as the Intel Nuc with its x86 architecture passing by price and board dimension advantages as in the LattePanda Delta3 and Raspberry PI4. In this section, we discuss our experience of running the AeroAssistant, particularly the segmentation pipeline described in Sec. 3.5.5 over the Voxl2 [61] a board that has dimensions comparable to a US quarter of a dollar (Figure 3.28) while capable of delivering 15 TOPS thanks to the SoC Qualcomm QRB5165 [88], which features the Neural Process Unit (NPU) for accelerated neural network inference which require further adjustments compared to a Cuda compatible GPU. The focus will then shift to the adjustments

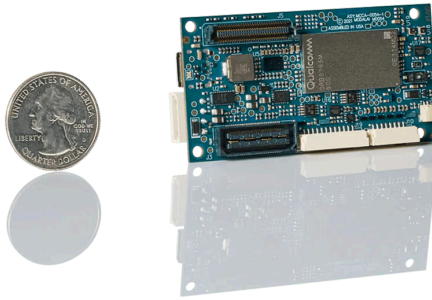


Figure 3.28. Voxl2 board [61]

necessary to run deep learning models on embedded hardware, as the core of AeroAssistant has been tested on LattePanda Delta3, Intel Nuc, Nvidia Jetson Orin Nano and Voxl2 without requiring any modifications to the framework. This demonstrates the capability to maintain a satisfactory control frequency for the UAV and provide a 30Hz video feedback to the operator.

Researchers have worked to improve the real-time performance of deep learning models using various techniques. Among the most common is distillation [89], which involves training a smaller, more specialized network to learn from a larger, typically more resource-intensive or general model [90]. Other approaches attempt to alter the architecture of the model to achieve the same task. For example, FastSam [68] modifies the underlying architecture compared to the original SAM [64] by employing instance segmentation, similar to YOLO-ACT [91], followed by a hard-coded segmentation selection. This modification, in contrast to the encoder-decoder architecture of SAM, achieves significantly faster performance. Furthermore, techniques such as Neural Architecture Search (NAS) [92–94] aim to automate the search for network architectures based on various criteria, including network accuracy, inference time, and network size. These approaches iteratively adjust the network until a Pareto-optimal network meeting the specified requirements is found. The final common approach is quantization [95], which is also the one we are going to deploy. This method involves initially training the network with high-precision weights, typically using *float 32*, then converting those weights to lower-precision equivalents such as *float 16* or *integer 8*. This process drastically reduces network size and inference time at the potential cost of accuracy loss, which might be negligible depending on the task and performance of the original network.

Depending on the chosen hardware platform for inference, it may be necessary to select a specific inference framework that supports the platform. Popular frameworks include PyTorch [96] and TensorFlow [97], which are interoperable thanks to the Open Neural Network Exchange (ONNX) [98], serving as a generic, interoperable, and industry-standard framework for runtime. Moreover, these popular frameworks have their optimized counterparts for mobile inference, namely ExecuTorch [99] and TensorFlow Lite [100], developed with efficiency in mind, featuring minimal memory footprint and operations suitable for microcontrollers.

In our case, the Voxl2 features a Tensorflow Lite inference engine capable of exploiting the NPU of the Qualcomm SoC. As previously mentioned in Sec. 2.4 for the implementation onboard FastSAM was used as it stands out for its tradeoff between accuracy and inference time. Therefore the original network is at first converted to ONNX and then translated to Tensorflow Lite using the tool `onnx2tf` [101] which is used to perform also quantization to *float 16*, this process allowed to obtain a inference time of $\sim 100\text{ms}$ which is sufficient to maintain a satisfactory interaction considering also the decoupled visualization and inference processes which hide the underlying computation maintaining a stable 30Hz video streaming. Before being usable, the raw retrieved output of the network underwent a Non-Maximum Suppression [102] to extract all predicted masks, which were then passed to the prompt encoder responsible for accurately segmenting the user-defined area.

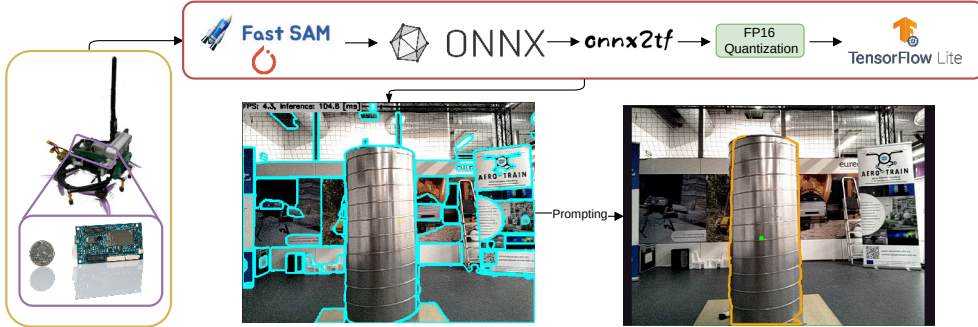


Figure 3.29. Network conversion Pipeline

3.8 Field Experiment

Within the context of the Aero-Train project, field experiments were conducted at the DTU Risø Campus. The task was to develop a platform capable of reaching a series of targets placed at various locations on the facility.



Figure 3.30. DTU Risø Campus Flying Area

The platform proposed to solve the challenge is depicted in Figure 3.31, an ex-copter equipped with a LattePanda Delta3 as the main computer. The sensors mounted were a Realsense D455 [63] for proximity depth estimation and RGB detection, and a Livox-Mid360 [103] used for reliably estimating odometry using Fast-LIO [104]. The solution features an actuated arm capable of extending the contact point by 20 cm.

To interact with the UAV, the AeroAssistant was used to identify the target and position the UAV in front of it before making contact. Thus, the plugins reported in Sec. 3.5.2 and Sec. 3.5.1 were connected and integrated with an autonomous target detection mechanism, simplifying further interactions with the UAV as depicted in



Figure 3.31. Drone Setup

Figure 3.32 while the autonomous s_r changes and integration with the target detector is reported in Figure 3.33.

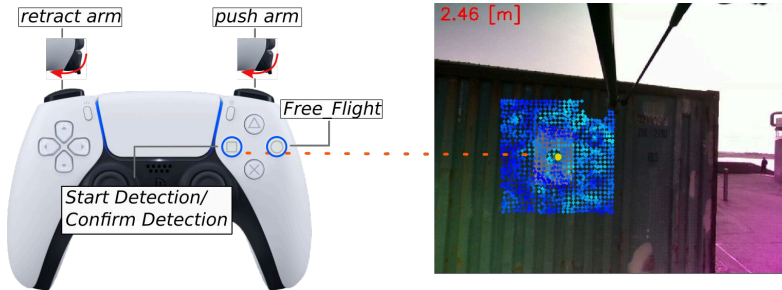


Figure 3.32. Interaction scheme

In this scenario, the operator initializes the process by pressing the square button, triggering the status $s_r = \textit{Aiming_To_Target}$. During this phase, the operator receives visual feedback of the detected target and the calculated path to reach it (Figure 3.34(a)). Once satisfied with the detection and path, the operator presses the square button again, changing the status to $s_r = \textit{Moving_To_Target}$, initiating the autonomous movement towards the target. Upon reaching the first estimated point, the status automatically changes to $s_r = \textit{Choose_Target}$ and waits for a satisfactory detection from the target detector. Once detected, the status changes to $s_r = \textit{Following_Target}$, initializing the KCF over the target and extracting n consecutive target points $\mathcal{P}^{ext} = \{\{\mathbf{p}_{d_1}, \mathbf{o}_{d_1}\}, \dots, \{\mathbf{p}_{d_n}, \mathbf{o}_{d_n}\}\}$. These points are then averaged in position and orientation to counteract potential noise from the target estimation.

$$\overline{\mathbf{p}_d} = \frac{\sum_{i=1}^n \mathbf{p}_i}{n} \quad (3.17)$$

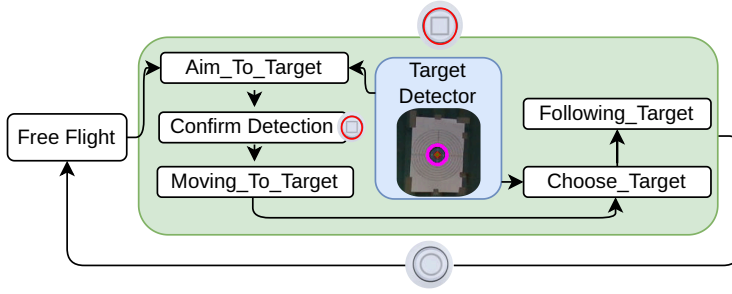


Figure 3.33. Autonomous detection and approach status changes pipeline

$$\overline{\mathbf{o}_d} = \frac{\sum_1^n \mathbf{o}_i}{n} \quad (3.18)$$

Finally, the average position $\overline{\mathbf{p}_d}$ and orientation $\overline{\mathbf{o}_d}$ are sent to the UAV to correctly position it in front of the target at the desired distance.

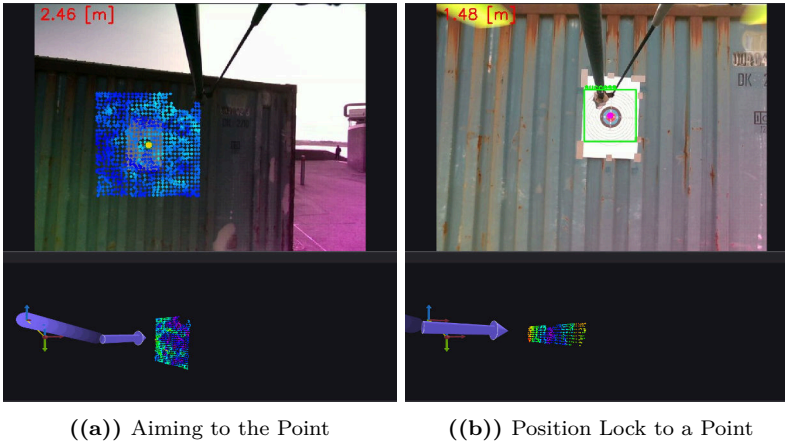


Figure 3.34. Operator Interface while Flying

Once the UAV reaches the desired point, the arm can be triggered using the left and right triggers to simplify the contact phase. Visualization of the extended arm is shown in Figure 3.35(a) from an external perspective and in Figure 3.35(b) from the operator's POV. Additionally, as shown in Figure 3.35(b), the distance to the closest point is displayed as text in the top-left corner of the interface, enhancing distance perception. This feature, made possible by the VisualizationManager and CloudManager, required minimal code adaptation to be displayed.

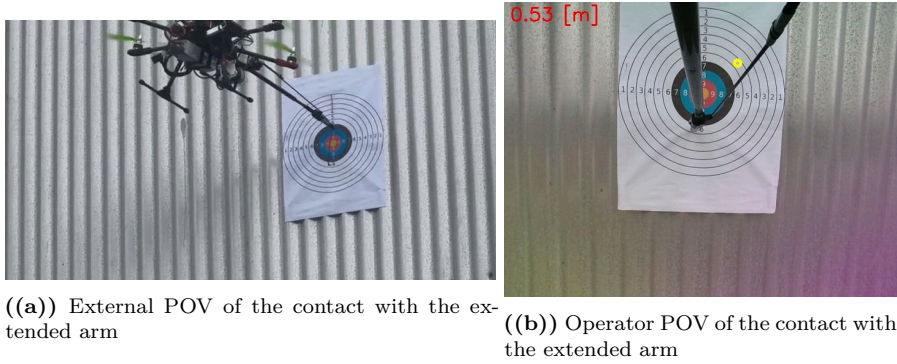


Figure 3.35. Extended arm in contact from different points of view

3.9 Bringing the Pilot into MR headsets

UAV teleoperation often relies on visualizing camera and spatial information on flat displays such as laptops, smartphones, or tablets. However, these displays suffer from inherent limitations due to their dimensions, restricting the amount of data that can be visualized. Additionally, outdoor operations can be hampered by sunlight reflecting off the display, degrading the visualization experience. This issue is usually mitigated either using a sun shield over the monitor, as shown in Figure 3.36(a) or for better immersion, as discussed in Sec. 1.4 VR goggles are also used. In this case, commercial solutions like DJI Goggles2 [49] are often used, particularly with FPV drones (Figure 3.36(b)), however, these solutions are tailored for specific drone configurations and do not support the visualization of spatial data or other sources of information beyond camera feedback.

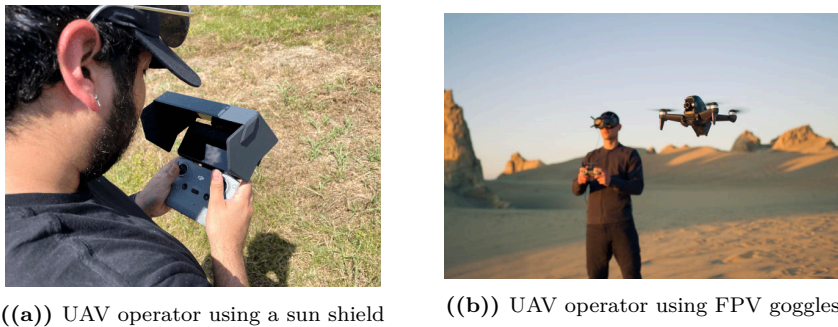


Figure 3.36. Operator Interface while Flying

Therefore, considering the objective of the AeroAssistant to enhance the UAV teleoperation experience, and given the recent advancements in commercial MR headsets

like Meta Quest 3 [105] and Apple Vision Pro [106], using an MR pilot interface to visualize information collected from the UAV appears to be a logical solution. This approach resolves spatial limitations that are no longer restricted by physical device dimensions and allows for the correct representation of spatial information, such as point clouds, without being affected by sunlight.

To facilitate this, as an initial step before developing a full teleoperation interface, we developed a communication layer that translates ROS1 [75] messages to ZMQ [77]. This layer is integrated with Unity3D [107] to ensure maximum interoperability across different headsets, independent of a specific ROS version on the interface side. A schematic representation of this communication layer is shown in Figure 3.37, which is responsible for establishing communication with the robot and abstracting various message types such as PointCloud, Text, and Camera, enabling easy integration with applications built on Unity3D that can run as stand-alone apps on MR devices.

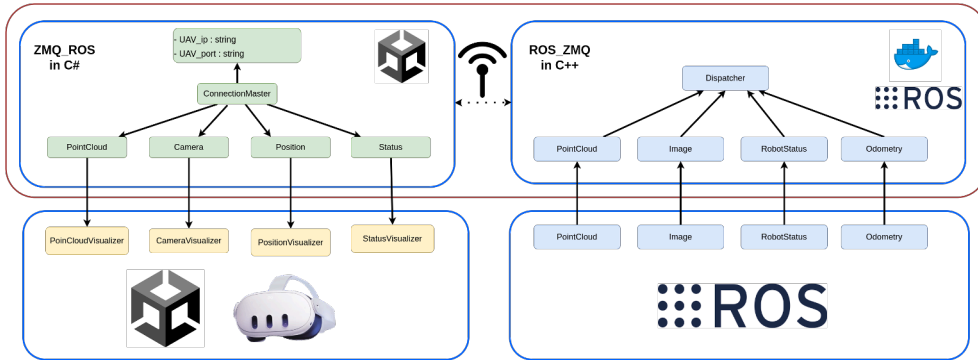


Figure 3.37. Schema passage from ROS to Unity

A mockup application was developed to test the bridge’s capabilities, demonstrating the ability to visualize images, point clouds, and interact with ROS by calling services within the headset Figure 3.38. In this setting a laptop was used to stream the image from the webcam while the pointcloud was generated as a series of points over a sphere.

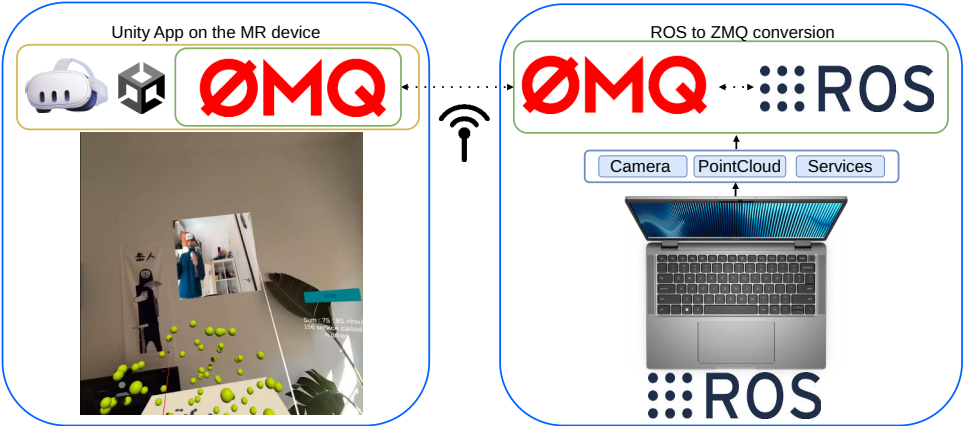


Figure 3.38. Mockup Visualization App

CHAPTER 4

Conclusion

This PhD research contributes to the field of assisted aerial teleoperation through the introduction of AeroAssistant, an innovative teleoperation framework that combines shared autonomy and augmented visualization to improve the teleoperation experience. Throughout the research, the proposed framework was tested in various environments, from controlled laboratories to full-scale mock-up industrial scenarios, and was executed on diverse platforms, ranging from hexacopters with manipulators to small, resource-constrained FPV-like drones. This experience highlighted the importance of having a flexible yet efficient framework capable of running on almost any modern computer. Through user studies, we validated parts of the framework, providing valuable insights into what is important when piloting a UAV and the significance of building a trust relationship between the operator and the UAV. These insights are crucial for ensuring that the features we develop are truly beneficial to the end user.

With the proposed work, the aim is to push the boundaries of human-drone interaction, delivering a tool that opens up possibilities for performing advanced operations and maneuvers without the need for expert pilots, while never compromising the feeling of control over the platform. However, despite the advances achieved throughout the PhD, numerous challenges still need to be addressed to enhance operations and achieve interactions that closely resemble the concept of a UAV as a companion rather than just a tool. In the subsequent section, the limitations of the system, along with possible research directions, are further evaluated.

4.1 Future Directions

Maintaining the idea of an interaction paradigm reminiscent of classical teleoperation mechanisms while incorporating advanced features, AeroAssistant has predominantly focused on a one-to-one relationship with the UAV, wherein an operator directly controls a single UAV with a RC controller through a flat display such as monitor or tablet. However there are different areas which deserve further investigation and exploration, such as multi UAV control (Sec. 4.1.1), usage of immersive displays (Sec. 4.1.2) for enhanced interaction, interfaces for assisted manipulation activities (Sec. 4.1.3) and interaction through natural language (Sec. 4.1.4).

4.1.1 Swarm Management

Depending on the nature of the operation, utilizing swarms of drones presents a promising approach to expedite tasks by distributing workload across multiple drones. Researchers have already begun exploring this direction by investigating the efficacy of user-centric interfaces for human-swarm teaming [108], as well as exploring body gesture control [109] and head-mounted displays for swarm tracking and relative localization [110]. Nevertheless, these interactions suffer from a lack of familiar interaction schemes and do not adequately address the need for seamless UAV control transition. Therefore, further exploration of interactions within a paradigm akin to the one currently proposed but extended to a multi-robot scenario could prove both intriguing and pertinent in facilitating swarm control while retaining the operator at the core of the decision-making process.

4.1.2 Immersive Interfaces

As already stated in Sec. 3.9 current teleoperation visualization devices are predominantly confined to flat screens, such as tablets or computer interfaces. This limitation is primarily due to their commercial availability, affordable prices, and portability. However, these devices have screens with constrained dimensions, which restrict the amount of information that can be effectively displayed. Additionally, they often suffer from light reflection issues, which can hinder the user experience.

Given the recent advancements in commercial XR devices, such as Meta Quest 3 [105], Apple Vision Pro [106], Pico 4 [111], Varjo XR4 [112], and HTC Vive Pro [113], there is now an opportunity to develop teleoperation control interfaces that are not constrained by screen dimensions. These XR devices offer the potential to create a more immersive camera streaming experience and provide better representation of spatial data, thereby offering immediate feedback to the operator. Several studies, such as [45–47], have already begun to explore the utilization of immersive devices for remote teleoperation and manipulation tasks. Consequently, implementing immersive displays for AeroAssistant appears to be a logical progression to further enhance and streamline the teleoperation experience.

4.1.3 Assisted Manipulation

Another aspect to take in consideration for further research is about the recent advancements in UAV manipulation capabilities [114], as evidenced by UAVs capable of tasks such as drilling [115], locking to pipes [116], and possessing full compliant arms [117], it is logical to consider how these interactions could benefit from enhanced interaction through shared autonomy paradigms. In such paradigms, the operator sends high-level commands while the UAV maintains the operator informed through an augmented interface, effectively acting as a supervisor throughout the operation.

Additionally, incorporating haptic feedback could significantly enhance the operator's ability to receive non-visible feedback during these operations.

4.1.4 Natural Language Interaction

One crucial aspect that could potentially drastically alter the interaction with UAVs is through natural language. In recent years, with the discovery of the effectiveness of Reinforcement Learning with Human Feedback [118] in controlling language models, we have witnessed an explosion of Large Language Models (LLMs) [119, 119–122] with remarkable reasoning capabilities and the ability to answer even the most complex questions. Paired with the most advanced speech recognition models [123], this development could pave the way for a more natural interaction, as showcased in [124] with ChatGPT. Within the framework of AeroAssistant, a very promising direction to explore could lie in building a natural interaction scheme where the operator issues high-level commands such as "Take Off" or "Go towards that Pipe," and the UAV generates a plan [125, 126] that could be displayed in an interface similar to [127] where the operator can control, accept, and modify potential actions, thereby ensuring that full control remains in the operator's hands.

Appendices

APPENDIX A

Articles

The following is a list of published and submitted work along the studies.

Learn to efficiently exploit cost maps by combining RRT^{*} with Reinforcement Learning

Riccardo Franceschini^{1,2}, Matteo Fumagalli² and Julian Cayero Becerra¹

Abstract—Safe autonomous navigation of robots in complex and cluttered environments is a crucial task and is still an open challenge even in 2D environments. Being able to efficiently minimize multiple constraints such as safety or battery drain requires the ability to understand and leverage information from different cost maps. Rapid-exploring random trees (RRT) methods are often used in current path planning methods, thanks to their efficiency in finding a quick path to the goal. However, these approaches suffer from a slow convergence towards an optimal solution, especially when the planner’s goal has to consider other aspects like safety or battery consumption besides simply achieving the goal. Therefore, we propose a sample-efficient and cost-aware sampling RRT^{*} method that is able to overcome previous methods by exploiting the information gathered from map analysis. In particular, we leverage the use of a Reinforcement Learning model able to guide the RRT^{*} sampling towards an almost optimal solution. We demonstrate the performance of the proposed method against different RRT^{*} implementations on multiple synthetic environments.

I. INTRODUCTION

Robot path planning tries to solve the task of finding an optimal path without collisions between two points in an environment where the optimality is defined by a cost function [5]. Multiple path planning algorithms has been developed such as A^{*} [8] or APF [14] methods. However, these methods either tend to be inefficient or, suffer from local minima. For these reasons, the whole family of Rapid-exploring Random Trees (RRT [16]) and its variants such as RRT^{*} [13] and Informed RRT^{*} [7] have become very popular in path planning due to their ability to quickly find a path combined with asymptotic optimality.

Depending on the task, various aspects may need to be considered when planning. For example, in an infrastructure inspection scenario, avoiding risky or complex areas could be crucial [19], similarly, it could be fundamental to plan according to the forecasted human position in search and rescue scenarios [9] or consider social aspects while navigating between humans [22]. In this regard, Lu et al. [20] showed the importance and difficulty of effectively combining multiple cost maps when planning. Thus, being able to efficiently find a path and safely navigate to complex and socially complex environments is still an open challenge even in 2D, and researchers in the literature have proposed multiple approaches addressing different aspects such as [23] and [4]

where they developed reinforcement learning agents capable of producing risk-aware policies for navigating in complex environments with dynamic obstacles such as humans. Still, those methods are focused on navigation rather than planning, hence, they are not directly related to the problem we are facing. Also relevant are [24, 2, 3], where the concept of risk is introduced to avoid collision with humans. In this case, the RRT planner estimates the probability of collision and constrains the path to maintain socially accepted distances from people. Huan et al. [10], proposed a Soft Actor Critic model capable of generating paths based on a previously defined maximum accepted risk. Also, in [18] and [28] the authors proposed models capable of predicting a probability distribution to guide RRT sampling from a data set of already resolved maps. While Choi et al. [11] developed a sample efficient RRT planner using Q-Learning for node selection for a robotic arm planning scenario. However, the methods described above do not directly consider the use of cost maps in their formulation and therefore are not directly comparable with our formulation. On the other hand, methods such as T-RRT[12] and T-RRT^{*}[6] encode cost maps in their sampling strategy accepting or rejecting new nodes depending on their cost, in particular T-RRT^{*}[6] will be considered in the evaluation phase.

We propose in this article a novel way of combining reinforcement learning with RRT^{*} to efficiently leverage information from cost maps. Our approach differs from other methods because it learns how to combine multiple cost maps by understanding where to sample without the need to pre-generate optimal trajectories such as [18] and [28]. We are also able to keep the RRT^{*} approach and not lose asymptotic optimality such as [10].

The remaining of this paper is structured as follows. In Sec.II, problem II-A, map generation II-B, baselines II-C and RL-RRT^{*} II-D are introduced. Then, in Sec.III our method is evaluated against the baselines, also comparison of visual paths and trees is proposed.

II. METHOD

A. Problem Setting

Following the definition of a path planning problem as in [28], let $X \in \mathbb{R}^n$ represent the state space of all possible robot configurations. Let X_{free} and X_{obs} represent the portions of the state space that can be achieved and unreachable, respectively, given the presence of obstacles or map boundaries. A discrete path, composed by an ordered collection of states $\phi = [x_0, x_1, x_2, \dots, x_k]$ is considered a solution of the path

¹Eurecat, Centre Tecnològic de Catalunya, Robotics and Automation Unit, Cerdanyola del Valles, Barcelona, Spain. riccardo.franceschini@eurecat.org
julian.cayero@eurecat.org

²Automation and Control group, Department of Electrical Engineering, Danish Technical University, Elektrovej, Building 326, DK-2800 Kgs. Lyngby Denmark mafum@elektro.dtu.dk

planning problem if $\forall i \in \{0, 1, 2, \dots, k-1\}$ and $\forall \gamma \in [0, 1]$

$$\begin{aligned} (1-\gamma)x_i + \gamma x_{i+1} &\in \mathcal{X}_{free}, \\ (1-\gamma)x_k + \gamma x_{goal} &\in \mathcal{X}_{free}, \\ \|x_k - x_{goal}\| &< r, \end{aligned} \quad (1)$$

with r representing a preset threshold.

Path solutions are not unique in general, hence letting Φ to represent all the possible solutions, i.e., all the possible discrete paths ϕ_i subject to (1), a big portion of path planner algorithms are designed to provide $\phi^* \in \Phi$, a viable path that minimize a given cost function $c(\phi)$:

$$\phi^* = \arg \min_{\phi \in \Phi} c(\phi). \quad (2)$$

In this work, we restrict the state space to \mathbb{R}^2 , and consider two cost functions. The J_l defined as

$$J_l = \sum_{i=0}^{k-1} \|x_i - x_{i+1}\| \quad (3)$$

is the path length, which give raise to the traditional RRT* planner and the second one is J_c

$$J_c = \sum_{i=0}^{k-1} \sum_{j=0}^p C \left(x_i + jd \frac{(x_{i+1} - x_i)}{\|(x_{i+1} - x_i)\|} \right) \quad (4)$$

that is the accumulated cost of the path, d is the step size,

$$p = \text{floor}(\|(x_{i+1} - x_i)\| / d) \quad (5)$$

represents the amount of steps between adjacent nodes and $C(x)$ represents the cost of the state x computed as the spatial average of the nearest obstacle distance for the given position and its surroundings. Hence the two are combined in

$$J(\phi) = J_l + \lambda J_c \quad (6)$$

where λ in Eq. 6 plays the role of a tuning factor enabling a trade-off between the two presented costs.

B. Map Generation

Similarly to [18], we built a dataset of randomly generated maps starting from 3 obstacles primitives respectively walls, cages and single obstacles (Fig. 1). By combining the 3 ob-

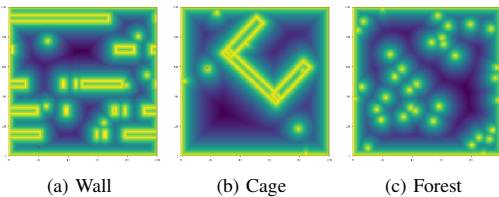


Fig. 1: Obstacle primitives

stacle primitives, 4000 random maps of dimension $[100, 100]$ were generated, equally distributed among the possible map combination, as shown in Table I.

Among the generated maps 80% are used for the training phase, while 20% remains for validation.

Obstacles	Environment					
	1	2	3	4	5	6
Vertical wall	✓				✓	
Horizontal wall		✓				✓
Cage			✓			
Forest			✓	✓	✓	✓

TABLE I: Environment categorization

1) *Distance Map*: Given a map, for each point on the map, the distance to the nearest obstacle is calculated using a k-d tree. Then, the cost is defined as the obstacle distance average over the n nearest point. In our case, a 3×3 spatial filter [25] is used to efficiently find the mean.

C. Baselines

This subsection presents the algorithms that will be used as baseline to asses the performance of the new approach presented in Sec. II-D. As first baseline we implemented RRT* from [26] and used as non cost-aware baseline. Then the distances similar to [17] are considered in the cost function as in (6), defined from now on as RRT*-c. However, RRT* randomly samples in the \mathcal{X} state space, doing so uses many iterations to explore parts of the map that are not useful for a low-cost solution. Thus bringing the approach, to ineffective sampling and slow convergence to the optimal path. Hence, the first approach developed was to modify the node sampling distribution by treating the distances values as distribution to guide the sampling like [1]. Thus, the probability of retrieving a node $x_j \in \mathcal{X}$ is

$$P(x_j) = \frac{C(x_j)}{\sum_i^{\mathcal{X}} C(x_i)}. \quad (7)$$

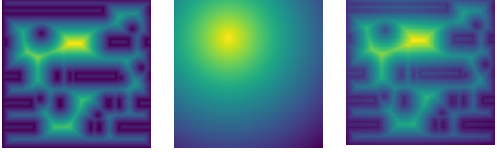
From now on we will refer to this approach as RRT**single.bias*. However, RRT**single.bias* does not take in consideration the direction to the goal while sampling. To address such problem, the goal distance map Fig. 2b and distance map Fig. 2a are combined as shown in Fig. 2c and used as probability distribution in the sampling process with brighter points being more likely to sample. We will refer to this approach as RRT**multiple.bias*. The probability function is then defined as:

$$P(x_j) = \frac{C(x_j) + \|x_j - x_{goal}\|}{\sum_i^{\mathcal{X}} C(x_i) + \|x_i - x_{goal}\|} \quad (8)$$

Also T-RRT* [6] is implemented, here the authors propose a transition function that through the usage of an adaptive temperature T (in our case $T_{init} = 10^{-6}$ and $T_{rate} = 0.1$) accept or reject new nodes depending on the difference of cost with respect to the node that are already in the three. For all the methods listed that make use of λ , a value of $\lambda = 100$ has been chosen. This particular values has demonstrated to give enough emphasis to the safety without over-unbalancing the relationship between distance and safety in the cost equation.

D. RL-RRT*

This section introduces the RL-RRT* method and the implementation of the collaborative framework that connects the RL agent and the RRT* algorithm.



(a) Distance (b) Goal (c) Goal + Distance
Fig. 2: Different sampling distributions

Initially, the sampling region is restricted to a square around a selected node represented by the black square and the green point in Fig. 4 (*planner_sampling_area*). The action space of the RL agent is defined over the sampling region as the four possible partitions of the original square named as 0, 1, 2 and 3 areas and representing the direction where to sampling next defined as *agent_sampling_area*.

Then, RRT* samples n random nodes following (8). The closest node to the center of the selected sampling area is then selected as the reference node, the sampling area is updated to its surroundings and the process is repeated until the goal or the maximum iteration are reached. Algorithm. (1) summarize the process and Fig. 3 draft the network and connections.

1) *Implementation details*: To understand the environment and take decision the model receive $n * global_map$, $n * local_map$ and the state defined as $state = [p, c, d, i]$. Where *global_map* is the complete map is enriched with *best_node* (green), *start_node* (red) and *goal_node* (blue) as in Fig. 5. While *local_map* is the visual representation of the *planner_sampling_area* previously described (Fig. 4). The number of cost maps is described by n (in our case $n = 2$, the distance map (Fig.2a) and the goal map (Fig.2b)). Thus, both visual enriched representations are converted to grayscale and resized to ensure efficient performance without losing relevant information. Regarding the state vector, $p = (x, y)$ is the current *best_node* position, c is the cost over distance value from *start_node* to *best_node*, d is the distance to the goal and i is *current_iter/max_iter*. Maps are processed through two separate Visual Encoders as presented in Fig. 3. These encoders are a combination of two convolutional layers followed by linear layers. Both local and global encoders have the same structure but they differ in the size of the first convolutional kernels. The global encoder has a larger kernel (5×5) for extracting global features while the local encoder has a smaller kernel (3×3) to focus more on local features. Subsequently, the extracted characteristics are processed through linear layers which have the task of distilling the information by reducing the dimensionality before the concatenation with the other state values. Then, the concatenated features are processed through the fusion layer which consists on a combination of multiple linear layers. The final output, are the action probabilities used to create a categorical distribution from which the action is sampled and passed to the environment. The agent was trained using PPO [27] due to its data efficiency and reliability, with actor and critic learning rate respectively $lr_actor = 0.0003$ and $lr_critic = 0.001$, as optimizer

Adam [15] and loss MSE (Mean Square Error). Furthermore, to maintain the asymptotic optimality of RRT*, we allow the planner to explore regions outside *agent_sampling_area* by not constraining an $r\%$ of the total amount of samples to the agent action.

A dynamic r that depends on the iteration number has been considered as

$$r(q) = \begin{cases} 0, & \text{if } q < aI_{max} \\ \alpha^{(q-aI_{max})}, & \text{if } aI_{max} \leq q \leq bI_{max} \\ Th, & \text{otherwise} \end{cases} \quad (9)$$

with $0 < a < b \leq 1$, Th representing the maximum percentage of samples not in the *agent_sampling_area*, I_{max} representing the maximum number of iterations and $\alpha = Th^{(\frac{1}{(b-a)I_{max}})}$ a parameter selected to reach Th at bI_{max} . This choice allows the agent to fully control the sampling for the first iterations and include potential solutions of the RRT* *multiple_bias* as the iterations increase. Learning and validation results present in this paper are generated with $a = 0.5$, $b = 5/6$, $Th = 0.8$ and $I_{max} = [50 \dots 2000]$

Since the goal is to reduce the cost of the path while being efficient, we reward the agent only when a path is found according to (10) where n_iter is the number of iterations needed to find a path.

$$reward = \frac{J_l}{J_c} + \frac{1}{n_iter} \quad (10)$$

Algorithm 1 Environment Step

```

1: procedure ENV.STEP(action)
2:   % get the sampling area from the agent action
3:   boundaries  $\leftarrow$  env.get_sampling_bound(action)
4:   % perform n step of RRT* within the boundaries
5:   state, reward, done  $\leftarrow$  env.plan_n_step(boundaries)
6:   % return the information to the agent
7:   return state, reward, done
8: end procedure

```

III. RESULTS

This section presents the comparison of RL-RTT* against the baseline planners presented in Sec. II-C. The methods are evaluated over 800 randomly generated maps not present in the training data set of our method with maximum possible iterations ranging from 50 up to 2000 steps. Since the aim is to show the effectiveness of our approach in effectively find low-cost path. We have decided to let the baselines reach the maximum iteration possible, this allows the baselines to take full advantage of all possible iterations and find the best path based on their approach. Contrary, our RL-RTT* approach stops as soon as it finds a solution and the planner stops at the maximum iteration only if it doesn't find a solution. We decided to use this approach to further demonstrate our method's ability to effectively recover non-complex pathways on the first try.

For each map, method and iteration, an average of the success rate, path length, and the normalized cost over 3

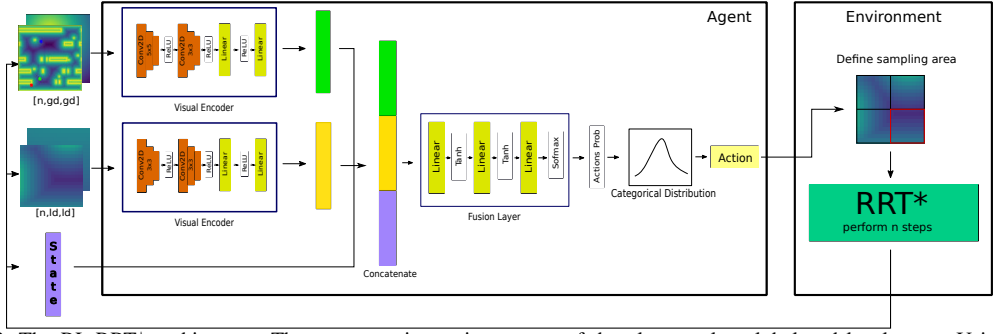


Fig. 3: The RL-RRT* architecture. The agent receive as input state of the planner plus global and local maps. Using this information the agent drives the RRT* multi_bias sampling by providing one of the 4 directions, then n nodes are sampled and the loop is repeated until a solution or the maximum iteration is reached.

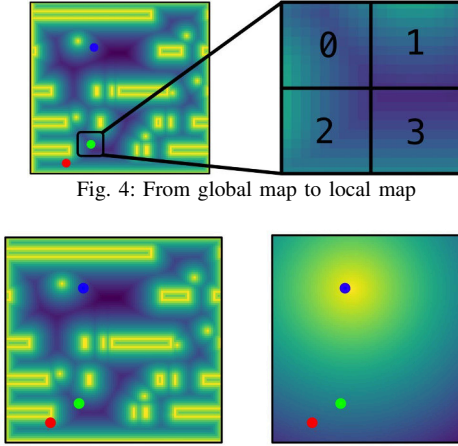


Fig. 4: From global map to local map

Fig. 5: Distance Map [left] Goal Map [right] with start goal and best nodes

separate tests, with different seeds in the random processes, are run to mitigate the possible effect of edge cases. The results provided are the success rate, defined as

$$Sr = \frac{N_{success}}{N_{tot}}, \quad (11)$$

J_c/J_l and J_l (Fig. 6). With $N_{success}$ and N_{tot} representing the number of successful experiment and the total number of experiments respectively.

The experiments where executed on a NVIDIA-GTX 1080ti with Python 3.8.8 and Pytorch [21] 1.10.2.

We want then evaluate the performance of our method against the other approaches. Fig. 6 reports the performances of the previously described baselines against RL-RRT*. As can be seen in Fig. 6, the number of maximum iterations plays an important role in performance. In particular, for a low value of maximum possible iterations, our method and the RRT with bias constantly struggle to find a solution even if those they find are significantly lower in terms of path cost (Fig. 6). We believe that these results are caused by the

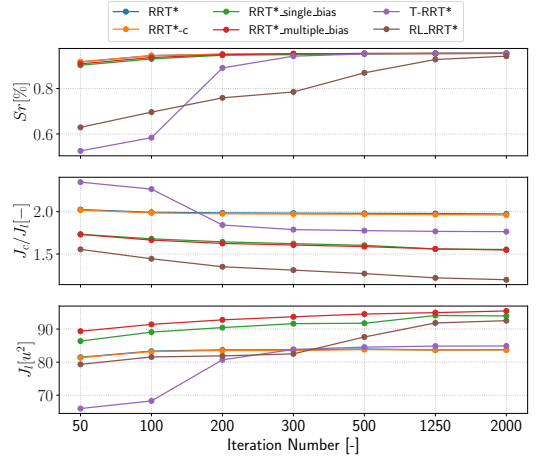


Fig. 6: Performance Comparison

further difficulty introduced in restricting sampling to less complex areas. This approach leads to limiting the set of admissible solutions to less complex solutions but at the same time reduces the set of all possible solutions by increasing the number of iterations necessary to constantly find a path.

Hence, considering an higher number of iterations the RL-RRT* considerably outperforms both standard RRT* and RRT* biased-sampling approaches in terms of Cost over path length while maintaining a similar Success Rates and distances that are in between to the ones found by the T-RRT* and the RRT* biased-sampling. Numerical results over the 2000 iterations test are reported in Table II that shows how the agent is capable of redirecting the planner towards non complex region without increasing too much the length of the path as it happens in the sampling based approaches. In addition, the average number of iterations required to find the path is reported which shows the ability of RL-RRT* to quickly find a non-complex path.

To further evaluate the performance and the behaviour of the different algorithms, we perform a case study on visual

Method	Sr(%)	J_l/J_d	J_d	Iter
RRT*	95.50	1.97	83.72	2000
RRT*-c	95.50	1.96	83.68	2000
RRT*single_bias	95.46	1.55	93.96	2000
RRT*multiple_bias	95.50	1.55	95.46	2000
T-RRT*	95.50	1.76	84.83	2000
RL-RRT*	94.25	1.20	92.53	319

TABLE II: Performance over 2000 iterations

examples (Fig. 7) with a maximum iteration number of 500. Comparing the tree evolution and final path generated by RRT*-c, T-RRT* and RL-RRT* approaches.

From Fig. 7 the differences between the approaches are worth to be further analyzed. In particular, it is interesting to note the differences between the classic RRT* (Fig. 7a and Fig. 7b) and RL-RRT* (Fig. 7e and Fig. 7f) paths and trees. The former randomly samples around the map ending up with a path that goes to the less safe narrow passage while the RL agent is able to make long-term decisions by guiding the tree's development to the safe path. Likewise, T-RRT* approach also limits the tree expansion thanks to the transition test, resulting in a sparse tree but with a safer path (Fig. 7c) than the one produced by the standard RRT*, however, is not as optimal as the RL-RRT* path.

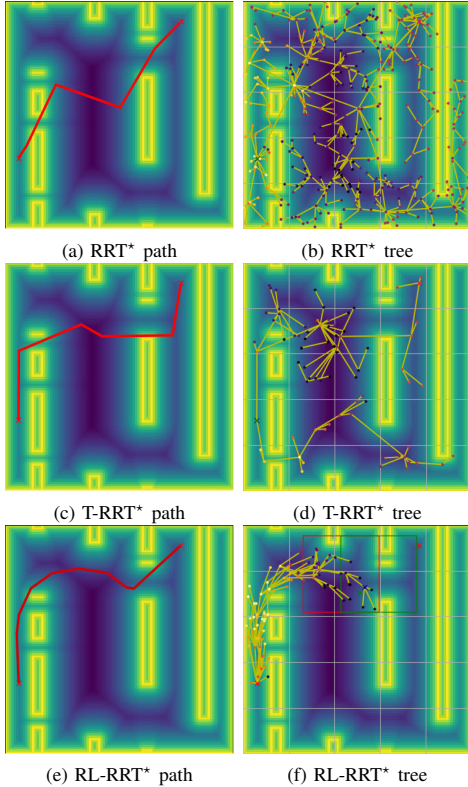


Fig. 7: Paths and Trees Comparison (500 Iterations)

IV. CONCLUSION

This work introduces a novel sample-efficient planner that mixes a reinforcement learning agent and RRT* solutions to find viable paths when considering several costmaps. The derived approach has been compared against different baselines in multiple synthetically generated environments considering path length and obstacle distance as objectives. The results derived demonstrated that RL-RRT* consistently outperforms the other approaches in finding the less complex path while maintaining good success rates and path length. Future work will extend the presented concept to the 3D case, targeting uncertain maps and considering noisy and imperfect sensor data. Moreover, a generalization of the current approach from 2 to n different costmaps will be proposed.

ACKNOWLEDGMENT

This work has been supported by the European Unions Horizon 2020 Research and Innovation Programme AERO-TRAIN under the Grant Agreement No. 953454.

REFERENCES

- [1] Bianca Bendris and Julián Cayero Becerra. “Design and Experimental Evaluation of an Aerial Solution for Visual Inspection of Tunnel-like Infrastructures”. In: *Remote Sensing* 14.1 (2022). ISSN: 2072-4292. DOI: 10.3390/rs14010195.
- [2] Wenzheng Chi and Max Q.-H. Meng. “Risk-RRT: A robot motion planning algorithm for the human robot coexisting environment”. In: *2017 18th International Conference on Advanced Robotics (ICAR)*. 2017, pp. 583–588. DOI: 10.1109/ICAR.2017.8023670.
- [3] Wenzheng Chi, Jiankun Wang, and Max Qing-Hu Meng. “Risk-Informed-RRT*: A Sampling-based Human-friendly Motion Planning Algorithm for Mobile Service Robots in Indoor Environments”. In: *2018 IEEE International Conference on Information and Automation (ICIA)*. 2018 IEEE International Conference on Information and Automation (ICIA). Aug. 2018, pp. 1101–1106. DOI: 10.1109/ICInfA.2018.8812396.
- [4] Jinyoung Choi et al. “Risk-Conditioned Distributional Soft Actor-Critic for Risk-Sensitive Navigation”. In: *CoRR* abs/2104.03111 (2021). arXiv: 2104.03111. URL: <https://arxiv.org/abs/2104.03111>.
- [5] Howie Choset et al. *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.
- [6] Didier Devaurs, Thierry Siméon, and Juan Cortés. “Optimal Path Planning in Complex Cost Spaces With Sampling-Based Algorithms”. In: *IEEE Transactions on Automation Science and Engineering* 13.2 (2016), pp. 415–424. DOI: 10.1109/TASE.2015.2487881.

- [7] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. "Informed RRT*: Optimal Incremental Path Planning Focused through an Admissible Ellipsoidal Heuristic". In: *CoRR* abs/1404.2334 (2014). arXiv: 1404.2334. URL: <http://arxiv.org/abs/1404.2334>.
- [8] Peter E Hart, Nils J Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107.
- [9] Larkin Heintzman et al. "Anticipatory Planning and Dynamic Lost Person Models for Human-Robot Search and Rescue". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 8252–8258. DOI: 10.1109/ICRA48506.2021.9562070.
- [10] Xin Huang et al. "Risk Conditioned Neural Motion Planning". In: *arXiv:2108.01851 [cs]* (Aug. 4, 2021). arXiv: 2108.01851. URL: <http://arxiv.org/abs/2108.01851> (visited on 01/18/2022).
- [11] Jinwook Huh and Daniel D. Lee. "Efficient Sampling With Q-Learning to Guide Rapidly Exploring Random Trees". In: *IEEE Robotics and Automation Letters* 3.4 (2018), pp. 3868–3875. DOI: 10.1109/LRA.2018.2856927.
- [12] Leonard Jaillet, Juan Cortes, and Thierry Simeon. "Transition-based RRT for path planning in continuous cost spaces". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 2145–2150. DOI: 10.1109/IROS.2008.4650993.
- [13] Sertac Karaman and Emilio Frazzoli. "Sampling-based Algorithms for Optimal Motion Planning". In: *CoRR* abs/1105.1186 (2011). arXiv: 1105.1186. URL: <http://arxiv.org/abs/1105.1186>.
- [14] Oussama Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *Autonomous robot vehicles*. Springer, 1986, pp. 396–404.
- [15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: <http://arxiv.org/abs/1412.6980>.
- [16] Steven M. LaValle. "Rapidly-exploring random trees : a new tool for path planning". In: *The annual research report* (1998).
- [17] Jinhan Lee, Charles Pippin, and Tucker Balch. "Cost based planning with RRT in outdoor environments". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2008, pp. 684–689. DOI: 10.1109/IROS.2008.4651052.
- [18] Zhaoting Li, Jiankun Wang, and Max Q.-H. Meng. "Efficient Heuristic Generation for Robot Path Planning with Recurrent Generative Model". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 7386–7392. DOI: 10.1109/ICRA48506.2021.9561472.
- [19] Sara Ljungblad et al. "What Matters in Professional Drone Pilots' Practice? Ann Interview Study to Understand the Complexity of Their Work and Inform Human-Drone Interaction Research". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. URL: <https://doi.org/10.1145/3411764.3445737>.
- [20] David V Lu, Dave Hershberger, and William D Smart. "Layered costmaps for context-sensitive navigation". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 709–715.
- [21] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems* 32. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: <http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- [22] Claudia Pérez-D'Arpino et al. "Robot Navigation in Constrained Pedestrian Environments using Reinforcement Learning". In: *CoRR* abs/2010.08600 (2020). arXiv: 2010.08600. URL: <https://arxiv.org/abs/2010.08600>.
- [23] Claudia Pérez-D'Arpino et al. "Robot Navigation in Constrained Pedestrian Environments using Reinforcement Learning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 1140–1146. DOI: 10.1109/ICRA48506.2021.9560893.
- [24] Jorge Rios-Martinez, Anne Spalanzani, and Christian Laugier. "Understanding human interaction for probabilistic autonomous navigation using Risk-RRT approach". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2011, pp. 2014–2019. DOI: 10.1109/IROS.2011.6094496.
- [25] Ashley Walker Robert Fisher Simon Perkins and Erik Wolfart. *Mean Filter*. <https://homepages.inf.ed.ac.uk/rbf/HIPR2/mean.htm>. 2003.
- [26] Atsushi Sakai et al. "PythonRobotics: a Python code collection of robotics algorithms". In: *CoRR* abs/1808.10703 (2018). arXiv: 1808.10703. URL: <http://arxiv.org/abs/1808.10703>.
- [27] John Schulman et al. "Proximal Policy Optimization Algorithms". In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347. URL: <http://arxiv.org/abs/1707.06347>.
- [28] Jiankun Wang et al. "Neural RRT*: Learning-Based Optimal Path Planning". In: *IEEE Transactions on Automation Science and Engineering* 17.4 (2020), pp. 1748–1758. DOI: 10.1109/TASE.2020.2976560.

Enhancing Human-Drone Interaction with Human-Meaningful Visual Feedback and Shared-Control Strategies

Riccardo Franceschini^{1,2}, Matteo Fumagalli² and Julian Cayero Becerra¹

Abstract—Recent developments in the capabilities of unmanned aerial vehicles (UAVs) have made them suitable for use in various industrial settings. Their ability to access difficult and remote locations, as well as providing remote manipulation and visual inspection capabilities, make them valuable for various industrial applications. However, operating UAVs can be challenging, particularly in cluttered environments. This research aims to enhance the teleoperation experience by providing human-meaningful information on the remote user interface, thereby improving the operator’s situational awareness. Shared autonomy routines utilizing the previously collected information are also developed to further assist the operator with challenging control tasks. The proposed system has been tested in simulated environments and on actual hardware.

I. INTRODUCTION

In recent years, unmanned aerial vehicles (UAVs) have steadily improved their ability to understand and move through complex environments. In doing so, considering also the emerging field of aerial manipulators [15], they have opened up a wide range of possible applications ranging from rescue operations [13, 1], visual inspection of civil infrastructure [3] to nondestructive testing (NDT) in industrial facilities such as bridges or Oil&Gas facilities [16]. However, these operations, in most cases, require an external operator who is usually located away from the UAV and is responsible for ensuring its safety. Therefore, maintaining a high level of pilot situational awareness is essential to ensure the success of the operation. Different approaches have been followed in the literature, [17, 5, 2, 14] have explored the use of the affordance primitives and visual cues to simplify remote control of the manipulator. In [10], the authors created a 3D visualization that represents a third point of view of the UAV interacting with the environment. While in [6, 7] they explored the use of Human-Embodiment for different manipulation activities. Also relevant is [11], in which an augmented reality (AR) interface is proposed from a third point of view to control an autonomous drone.

This work, similarly to [6, 9], proposes a human-drone interaction framework and is based on two insights. The first is that the operator currently receives only a portion of the information that the UAV is collecting; in fact, the pilots’ interfaces are mainly based on the analog camera stream with

some additional information such as gps position, battery status, ground elevation, or drone orientation, to name a few. However, this does not allow for real-time perception of what the drone is sensing, which is particularly important in cluttered environments to improve operator self-awareness, reducing the risk of dangerous maneuvers, and improving inspection quality. The second insight is that in order to retain full control over the UAV, the operator does not need to completely control the drone, but rather, only the decision-making regarding the drone’s actions is relevant. To address the first insight, a flexible structure is proposed for retrieving and combining pilot-relative information that can be displayed on the remote control device. This system adds a processing layer between the operator and the UAV to encode information about distances to obstacles, UAV odometry, orientation relative to surrounding surfaces, and uncertainty of sensor measurements. This enables real-time understanding of the UAV’s sensory data by the operator and allows for the selection of relevant information tailored to specific use cases. The second insight is addressed through the development of shared autonomy algorithms to align and follow the nearest surfaces. By doing so, the drone is responsible for avoiding dangerous maneuvers and maintaining the desired distance and attitude to the inspected surface, while the operator retains full control over the UAV.

Overall, this interaction framework aims to improve operator self-awareness and enhance the UAV piloting experience. The proposed method was evaluated in a simulated environment with various maps, demonstrating its effectiveness in maintaining the desired distance and orientation while following a facing surface.

II. VISUAL FEEDBACK

In the following sections, the methodology used for extracting human-meaningful information from sensor data is explained. It is assumed that the UAV is equipped with sensors capable of perceiving the environment in 3D and obtaining odometry data. First, the data preprocessing procedure is described, followed by an explanation of each of the extracted pieces of information.

A. Information Preprocess

Initially, the 3D sensory data is depicted in the form of point clouds. The augmented point cloud is characterized by $\mathbf{P} = \{\mathbf{x}_i | i = 1..N\}$, where each point $\mathbf{x}_i \in \mathbf{R}^3$ contains supplementary information denoted by k_i that is normalized within the range of $[0, 1]$. As a first step in reducing noise generated by the depth sensor, the received point cloud is

¹Eurecat, Centre Tecnològic de Catalunya, Robotics and Automation Unit, Cerdanyola del Vallès, Barcelona, Spain. riccardo.franceschini@eurecat.org
julian.cayero@eurecat.org

²Automation and Control group, Department of Electrical Engineering, Danish Technical University, Elektrovej, Building 326, DK-2800 Kgs. Lyngby Denmark mafum@elektro.dtu.dk

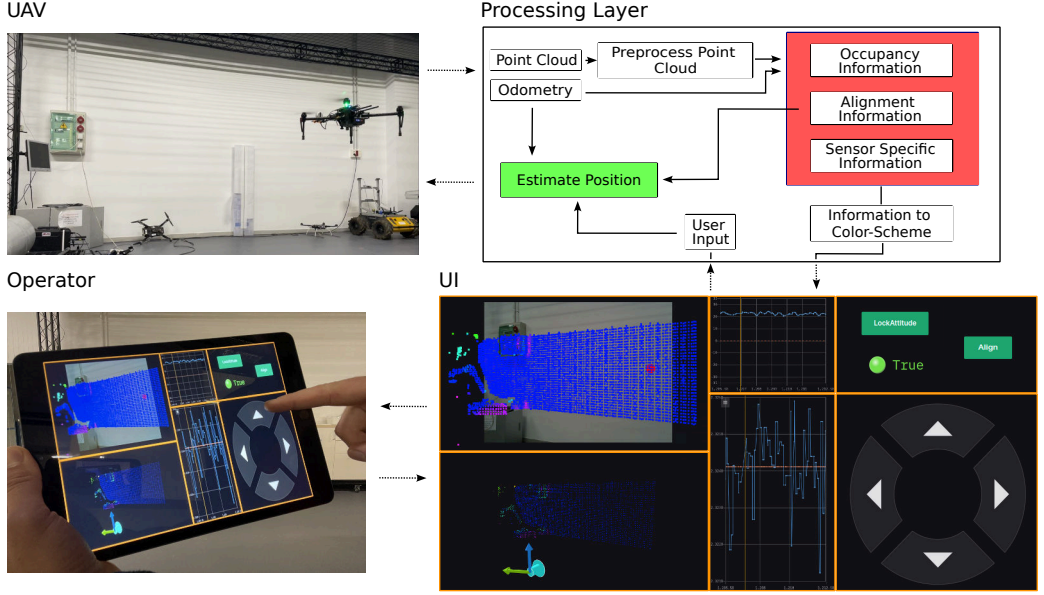


Fig. 1: This scheme represents the interaction between an operator and a UAV, where the operator uses a tablet to view information extracted from the processing layer and control the UAV by sending high-level commands. A possible remote control interface is shown in the bottom right corner. The upper left corner of the interface displays the video feedback from the UAV, with points sensed on the surface colored according to the UAV's alignment with the surface. The closest point, which is used as a reference for calculating alignment movement, is represented by a larger dot (red in the image) and its color indicates the angle difference with respect to the UAV's alignment. Possible colors are red, yellow, and green. The bottom left panel shows the 3D visualization of the UAV reference frame and the visualization of the colored point cloud with the previously described information. The graphs in the center show the current angle to the surface and the distance to the nearest point. On the right side of the panel, there are buttons for aligning the drone with the nearest surface, changing the drone's status to be aligned with the nearest surface, and moving while maintaining the correct attitude.

preprocessed through voxelization and removal of outliers using statistical and geometric filter methods. The geometric method discards points that have fewer than n points within a sphere of radius r , while the statistical method discards points that are further than a defined standard deviation σ from the average of their n nearest neighbors.

B. Alignment Information

In order to ensure safe remote operation of a UAV, it is essential to comprehend the UAV's alignment with respect to the surface it is facing, in addition to its current velocity and acceleration. Obtaining this information involves estimating the cloud normals of the surface using covariance analysis, which is represented by $\mathbf{n}_i = \{x_i, y_i, z_i\}$ for all $\mathbf{x}_i \in \mathbf{P}$. However, the covariance analysis algorithm may produce two opposite directions as normal candidates. To resolve this issue, the normals are aligned towards the drone using the camera origin as a reference point, resulting in consistently oriented normals as shown in Fig. 2.

Knowing the camera frame with respect to the UAV body frame, a direction vector $\mathbf{d} = [d_x, d_y, d_z]$ in camera frame can be defined representing the UAV orientation as in Fig. 3.

It is then possible to obtain $\forall \mathbf{x}_i \in \mathbf{P}$ the yaw γ_i offset by simply:

$$\gamma_i = \text{atan2}(\|\mathbf{n}_i \times \mathbf{d}\|, \mathbf{n}_i \cdot \mathbf{d}) \quad (1)$$

which due to the properties of atan2 and the $\|\mathbf{n}_i \times \mathbf{d}\|$ that is always positive returns angles that are $\gamma_i \in [0, \pi]$. The attitude angle θ_i is another important angle that is taken into consideration. Considering the UAV position $\mathbf{p} = [p_x, p_y, p_z]$ a vector \mathbf{t}_i representing the direction from the UAV center to each \mathbf{x}_i , is defined $\forall \mathbf{x}_i \in \mathbf{P}$ as :

$$\mathbf{t}_i = [p_x - x_i, p_y - y_i, p_z - z_i] \quad (2)$$

hence, the desired angle is:

$$\theta_i = \text{atan2}(\|\mathbf{t}_i \times \mathbf{n}_i\|, \mathbf{t}_i \cdot \mathbf{n}_i) \quad (3)$$

which like the yaw angle is $\theta_i \in [0, \pi]$. Thus, the alignment value normalized between $\psi_i \in [0, 1]$ is:

$$\psi_i = \frac{\theta_i + \gamma_i}{2\pi} \quad (4)$$

A value of $\psi_i = 1$ indicates a safe alignment, while a value of $\psi_i = 0$ indicates a non-safe alignment. It is important to

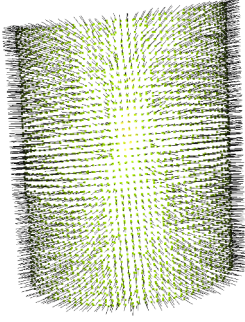


Fig. 2: Estimated normals of a pipe structure

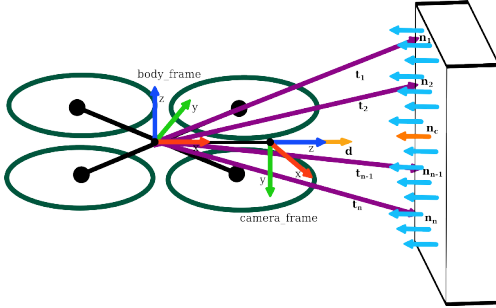


Fig. 3: Representation of UAV reference frames and vectors used to retrieve the alignment information

note that the alignment value does not distinguish between positive and negative values to avoid ambiguity in the displayed information. Additionally, the current angular and linear velocity of the UAV should be considered, affecting directly the alignment values. Therefore, angular Ω and linear velocity \mathbf{V} are extracted from the odometry and their magnitude $\|\mathbf{V}\|_2$, $\|\Omega\|_2$ is averaged over a time window of n data points:

$$\bar{\omega} = \frac{\sum_{i=0}^n \|\Omega\|_2}{n} \quad (5)$$

$$\bar{V} = \frac{\sum_{i=0}^n \|\mathbf{V}\|_2}{n} \quad (6)$$

to smooth the impact over the cloud visualization. Then, ψ_i is scaled as follow:

$$\psi_i = \psi_i^{1+\bar{\omega}+\bar{V}} \quad (7)$$

A visual example of the same surface with low and high velocities is reported in Fig. 4.

C. Occupancy Information

In order to ensure the accuracy and reliability of remote operations, it is essential to evaluate the confidence of the UAV in regards to its sensor data. To address this concern,

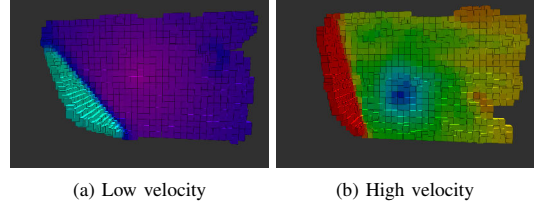


Fig. 4: Alignment Information at different linear and angular velocities

the use of HilbertMaps, as outlined in [18], can effectively facilitate the understanding of spatial data and identify potential outlier sensor readings. These maps utilize fast kernel approximations to transform point cloud data into a Hilbert space, where a logistic regression classifier is trained to evaluate the confidence of the sensor data. The probability of non-occupancy is defined as:

$$p(y_* = -1 | \mathbf{x}_*, \mathbf{w}) = \frac{1}{1 + \exp(\mathbf{w}^T \mathbf{x}_{i*})} \quad (8)$$

and respectively the occupancy probability is:

$$p(y_* = +1 | \mathbf{x}_{i*}, \mathbf{w}) = 1 - p(y_* = -1 | \mathbf{x}_{i*}, \mathbf{w}) \quad (9)$$

with \mathbf{x}_{i*} representing the sparse features obtained from the RBF projection of the original points \mathbf{x}_i :

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{x} - \mathbf{x}'\|_2^2\right) \quad (10)$$

and \mathbf{w} the vector that parametrizes the discriminative model. The advantage of this method lies in its ability to identify whether sensor noise is causing inaccurate results by providing the ability to obtain the probability of occupancy for each point, which can be easily interpreted by the operator using it. A visual example of the occupancy cloud is shown in Fig. 5, where the logistic regression can understand the noise and spatial relationship of the sensor point cloud by returning the occupancy probability, where dark colours indicate higher occupancy probability and light colours lower probability.

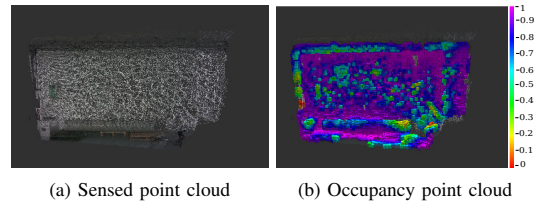


Fig. 5: On the left, the point cloud collected by the depth camera; on the right, the estimated probability of occupancy.

D. Sensor specific information

Depending on the sensor used, other information can be provided along with the point cloud. For example, the TOF sensor developed by STMicroelectronics [12] retrieves the

confidence, amplitude and ambient light values for each point. This information is usually discarded or used internally during data processing. However, this information can play an important role in increasing operator awareness and is therefore reported as cloud information. For example, in Fig. 6 the confidence is stable over the flat surface while it drops near the corners, or the ambient values show a darker pixel where the light changes.

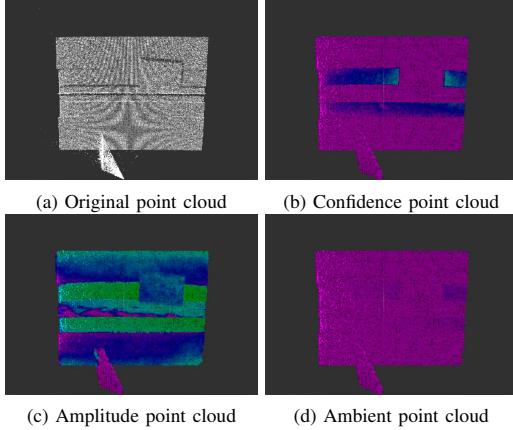


Fig. 6: Sensor Specific Information

III. SHARED AUTONOMY

This section propose the use of shared autonomy algorithms to enhance the piloting experience by allowing the operator to retain decision-making power while the movement control is fully offloaded to the UAV. These routines utilize previously collected data to understand and compute the desired movement and orientation. One routine aligns the UAV with the nearest surface, while the second routine maintains a constant attitude and distance to the nearest surface during lateral and vertical movements.

A. Align Routine

To align with the closest surface, the closest point \mathbf{x}_c is obtained, where the distance is defined by the euclidean distance from the UAV. Similarly to the alignment information previously retrieved (2), a vector \mathbf{t}_c that gives the direction from the UAV center to \mathbf{x}_c is used to find angle difference η :

$$\eta = \text{atan2}(\|\mathbf{d} \times \mathbf{t}_c\|, \mathbf{d} \cdot \mathbf{t}_c) \quad (11)$$

the sign s of the angle difference is retrieved analyzing the perpendicular component (y in the proposed frames in Fig.3) from the vector cross product:

$$s = \text{sign}((\mathbf{d} \times \mathbf{t}_c)_y) \quad (12)$$

The new orientation is retrieved by updating the UAV yaw ϕ :

$$\phi = \phi + s\eta \quad (13)$$

B. Surface Follow

To follow the closest surface, the operator set the UAV to the surface following mode, from that moment the UAV store internally the desired distance to maintain ξ . Then the operator is still in control of the lateral and vertical displacement with a controller like the one in Fig. 7. Thus,

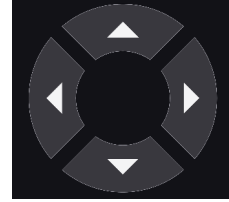


Fig. 7: Example of Controller

the new desired position is computed as follows. Given the command from the controller $\mathbf{c} = [l_c, v_c]$ with l_c, v_c representing lateral and vertical displacement with values $+/-1$ for positive or negative displacement and 0 for none. The current position $\mathbf{p}^t = [p_x^t, p_y^t, p_z^t]$ at time t is updated as:

$$p_y^{t+1} = p_y^t + \delta l_c \quad (14)$$

$$p_z^{t+1} = p_z^t + \delta v_c \quad (15)$$

with δ representing the desired offset. Then, given the new position \mathbf{p}^{t+1} a new closest point \mathbf{x}_c and its distance ξ_{x_c} is estimated using the current point cloud. The alignment angle η and vector to touch \mathbf{t} are retrieved. The angle will be updated as in (13) to maintain the alignment, while the position is updated to keep the desired distance.

$$p_x^{t+1} = p_x^t + (\xi_{x_c} - \xi)t_x \quad (16)$$

The updated position \mathbf{p} and orientation η is sent to the position controller.

Moreover, the color of the dot Fig. 1 reflects the safety of the contact following a familiar color scheme with green if $\eta < 5$, yellow if between $5 \leq \eta \leq 15$ and red otherwise.

IV. EXPERIMENT

In this section, the performance of the proposed shared-autonomy algorithm is evaluated on synthetic maps using RotorS [4] simulator in Gazebo [8] with realistic sensor noise simulation. The goal of these experiments is to assess the algorithm's ability to maintain the correct alignment and distance with respect to the facing surface under various map conditions and δ values.

Different maps are used in the experiments, representing different environments such as a vertical pipes and walls from inside. The evaluation consisted of starting from the same initial position and performing a loop around the desired shape, measuring the offset with respect to the desired distance of $\xi = 1.5$ meters and the angle with the closest point \mathbf{x}_c . The values were then compared under different δ values ranging from 0.1 meters to 0.5 meters. This

allows the evaluation of the algorithm's performance under a range of different conditions. Table I reports the average distance offset $\tau = \|\xi_{x_c} - \xi\|$ and angle offset η for each displacement δ across the tested environments.

$\delta[m]$	$\tau[m]$	$\eta[deg]$
0.1	0.037	13.52
0.2	0.071	15.40
0.3	0.072	15.31
0.4	0.089	14.75
0.5	0.071	14.41

TABLE I: Performance under different δ values

In figures 9, and 8, the trajectories with $\delta = 0.1$ m and $\delta = 0.5$ m are shown. The color scheme encodes the distance offset τ (Fig. 8b and 9b) and the angle offset η (Fig. 8a and 9a). As expected, with a smaller displacement of $\delta = 0.1$ m, the angle estimation and position are more reliable, producing smoother trajectories in which the UAV is more aligned with respect to the surface. This behaviour is due to the point cloud quality becoming worse as more distant points from the current position are considered, leading in this way to inaccurate attitude and position estimation. Thus, the final trajectories are more fragmented, such as in Fig. 8, where the UAV continuously adjusts its position and orientation to match the desired values. Relevant to consider is also the trajectory in Fig. 9 which shows that the drone gets very close to the surface near the corners before realizing that it should turn and continue in the other direction. This behavior is a result of the map-less approach, which relies on the current available information to compute the next point. Until the distance sensor detects the other part of the wall, the UAV will continue in the same direction, potentially resulting in a crash if the δ is too high. Possible solutions to this issue include using multiple depth cameras to provide a wider field of view and detect lateral obstacles, or implementing a safety mechanism that turns the drone to assess the safety of the movement. However, this option may affect the piloting experience as the drone continuously turns to analyze the environment. Overall the proposed method demonstrated its capability in maintaining the correct distance and attitude with different δ values, though with a reduced precision at higher δ values that may be tolerable depending on the task requirements.

The effectiveness of the proposed approach was verified on actual drone hardware in a controlled laboratory setting, as evidenced by Figure 1, successfully executing the computations on readily available commercial hardware. Nonetheless, when operating in real-world scenarios, the transmission of information can cause a considerable delay in the data visualization, which may negatively impact the user experience.

V. CONCLUSION

To conclude, this work introduces a new method for gathering, combining, and displaying multiple pieces of relevant information on a pilot's interface with the goal of

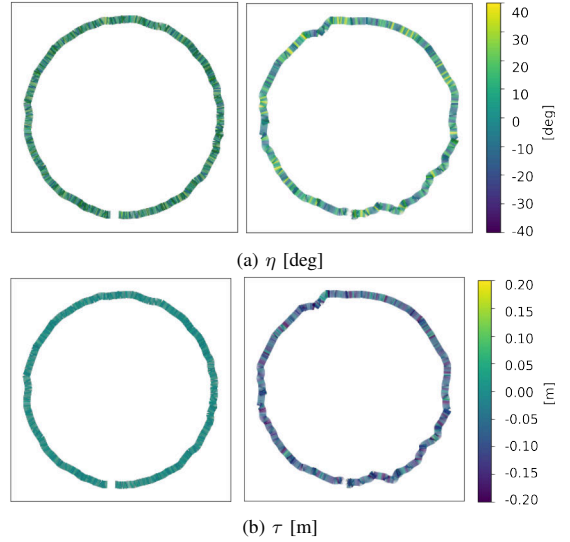


Fig. 8: Vertical Pipe, left $\delta = 0.1$ m, right $\delta = 0.5$ m

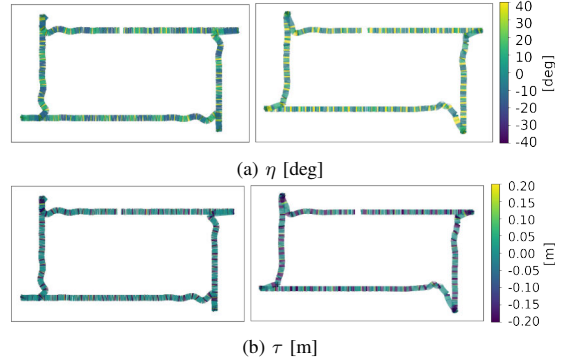


Fig. 9: Cube inside, left $\delta = 0.1$ m, right $\delta = 0.5$ m

improving situational awareness and reducing mental workload. In addition, the use of shared-autonomy schemes based on information obtained from the point cloud is proposed. The proposed method has been tested and validated in simulated and real-world environments, showing the ability to run on commercially available hardware. Future efforts will concentrate on enhancing the visualization experience with additional relevant information, implementing more advanced shared autonomy algorithms for complex tasks and reducing the visualization latency.

ACKNOWLEDGMENT

This work has been supported by the European Unions Horizon 2020 Research and Innovation Programme AERO-TRAIN under Grant Agreement No. 953454.

REFERENCES

- [1] Ankit Agrawal et al. "The Next Generation of Human-Drone Partnerships: Co-Designing an Emergency Response System". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, pp. 1–13. ISBN: 9781450367080. DOI: 10 . 1145 / 3313831 . 3376825. URL: <https://doi.org/10.1145/3313831.3376825>.
- [2] Stephanie Arevalo Arboleda et al. "Assisting Manipulation and Grasping in Robot Teleoperation with Augmented Reality Visual Cues". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: 10 . 1145 / 3411764 . 3445398. URL: <https://doi.org/10.1145/3411764.3445398>.
- [3] Bianca Bendris and Julián Cayero Becerra. "Design and Experimental Evaluation of an Aerial Solution for Visual Inspection of Tunnel-like Infrastructures". In: *Remote Sensing* 14.1 (2022). ISSN: 2072-4292. DOI: 10 . 3390 / rs14010195. URL: <https://www.mdpi.com/2072-4292/14/1/195>.
- [4] Fadri Furrer et al. "Robot Operating System (ROS): The Complete Reference (Volume 1)". In: ed. by Anis Koubaa. Cham: Springer International Publishing, 2016. Chap. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. ISBN: 978-3-319-26054-9. DOI: 10 . 1007 / 978-3-319-26054-9_23. URL: http://dx.doi.org/10.1007/978-3-319-26054-9_23.
- [5] Stephen Hart et al. "Generalized Affordance Templates for Mobile Manipulation". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 6240–6246. DOI: 10 . 1109 / ICRA46639.2022.9812082.
- [6] Dongbin Kim and Paul Y. Oh. "Aerial Manipulation using a Human-Embodied Drone Interface". In: *2022 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*. 2022, pp. 1–7. DOI: 10 . 1109 / ARSO54254.2022.9802972.
- [7] Dongbin Kim and Paul Y. Oh. "Toward Avatar-Drone: A Human-Embodied Drone for Aerial Manipulation". In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2021, pp. 567–574. DOI: 10 . 1109 / ICUAS51884.2021.9476704.
- [8] N. Koenig and A. Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10 . 1109 / IROS.2004.1389727.
- [9] Konstantinos Konstantoudakis et al. "Drone Control in AR: An Intuitive System for Single-Handed Gesture Control, Drone Tracking, and Contextualized Camera Feed Visualization in Augmented Reality". In: *Drones* 6.2 (2022). ISSN: 2504-446X. DOI: 10 . 3390 / drones6020043. URL: <https://www.mdpi.com/2504-446X/6/2/43>.
- [10] Jongseok Lee et al. "Visual-Inertial Telepresence for Aerial Manipulation". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 1222–1229. DOI: 10 . 1109 / ICRA40945.2020.9197394.
- [11] Chuhao Liu and Shaojie Shen. "An Augmented Reality Interaction Interface for Autonomous Drone". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 11419–11424. DOI: 10 . 1109 / IROS45743.2020.9341037.
- [12] MultiMedia LLC. *ST Microelectronics*. 2022. URL: https://www.st.com/content/st_com/en.html (visited on 09/15/2022).
- [13] Antonio Loquercio et al. "Learning high-speed flight in the wild". In: *Science Robotics* 6.59 (2021), eabg5810. DOI: 10 . 1126 / scirobotics . abg5810. eprint: <https://www.science.org/doi/pdf/10.1126/scirobotics.abg5810>. URL: <https://www.science.org/doi/abs/10.1126/scirobotics.abg5810>.
- [14] Levi Manring et al. "Augmented reality for interactive robot control". In: *Special Topics in Structural Dynamics & Experimental Techniques, Volume 5*. Springer, 2020, pp. 11–18.
- [15] Anibal Ollero et al. "Past, Present, and Future of Aerial Robotic Manipulators". In: *IEEE Transactions on Robotics* (2021). Conference Name: IEEE Transactions on Robotics, pp. 1–20. ISSN: 1941-0468. DOI: 10 . 1109 / TRO.2021.3084395.
- [16] Anibal Ollero et al. "The AEROARMS Project: Aerial Robots with Advanced Manipulation Capabilities for Inspection and Maintenance". In: *IEEE Robotics Automation Magazine* 25.4 (Dec. 2018). Conference Name: IEEE Robotics Automation Magazine, pp. 12–23. ISSN: 1558-223X. DOI: 10 . 1109 / MRA.2018.2852789.
- [17] Adam Pettinger et al. "Reducing the Teleoperator's Cognitive Burden for Complex Contact Tasks Using Affordance Primitives". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 11513–11518. DOI: 10 . 1109 / IROS45743.2020.9341576.
- [18] Fabio Ramos and Lionel Ott. "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent". In: *The International Journal of Robotics Research* 35.14 (2016), pp. 1717–1730. DOI: 10 . 1177 / 0278364916684382. eprint: <https://doi.org/10.1177/0278364916684382>. URL: <https://doi.org/10.1177/0278364916684382>.

Riding the Rollercoaster: easing UAV Piloting Experience with XR and continous planning

Riccardo Franceschini^{1,2}, Matteo Fumagalli² and Julian Cayero Becerra¹

Abstract—Operating unmanned aerial vehicles (UAVs) in complex environments can be challenging, particularly for inexperienced operators. This paper introduces a method aimed at enhancing the piloting experience by incorporating an intermediary processing layer between the remote controller and the drone. The proposed approach allows the operator to control the UAV's speed and direction along a safe path that is dynamically computed and visualized on the camera stream in an XR fashion. The UAV, autonomously plan and execute the path, while adapting to the operator's inputs and environmental changes. The main objective of the proposed solution is to improve the operator's situational awareness and perception, as well as the safety and efficiency of the UAV navigation. The paper outlines the system and methodology employed, showing its ability to operate at a high enough frequency to enable seamless user interactions.

I. INTRODUCTION

Teleoperation remains a prevalent mode of human-robot interaction for contemporary robotic systems, particularly for unmanned aerial vehicles (UAVs). The collaboration between the drones and the operators is crucial in ensuring safety in tasks that require precise navigation in cluttered and hazardous environments, as mentioned in [10, 1, 9]. However, teleoperating a drone in such scenarios can present significant challenges for novice users, who may have limited situational awareness and visual feedback when the UAVs are far from them. To enhance the user experience, various approaches have been proposed in the literature to enable the operators and the UAVs to function as a team, sharing information through extended reality (XR) interfaces and shared control strategies. For example, researchers have employed augmented reality and haptic feedback to improve the operators' perception in different aspects. Some have shown the robot's surroundings through immersive interfaces with the UAVs moving in a superimposed 3D representation of the environment [5, 13], while others have augmented information regarding the actions they were taking while remote piloting [3]. Additionally, some have mixed AR cues with haptic feedback in robot manipulation tasks [4]. Furthermore, other approaches have allowed the operators to modify parts of the drones' trajectory using haptic devices while leaving the UAVs to handle planning and execution [8, 7, 12]. Deep learning agents have also been employed to

infer the user's intentions from their inputs and use motion primitives to control the drone, while displaying suggested actions in augmented reality [15, 14]. This paper proposes a novel approach to share the control between a UAV and an operator. Instead of learning a trajectory or controlling the shape of a path, the approach allows the drone to compute a safe path. Meanwhile, the operator controls the direction and speed of the UAV along the path with a SONY DualSense™ Wireless Controller [2], and the experience is further enhanced with the usage of haptic feedback to represent drone velocity. This interaction framework named Rollercoaster, aims to improve the operator's situational awareness and facilitate the UAV piloting experience.

II. METHODOLOGY

This section explains the methodology used for planning and sharing the control over the bilateral control execution. The approach is sensor-agnostic; however, it assumes that the UAV can estimate its odometry and perceive its surrounding 3D environment, whether through depth cameras or LiDAR. First, the data planning and mapping procedure is described (Sec. II-A), then an architecture for the integration of the operator in the planning loop is proposed (Sec. II-B) and finally the visualization technique is described (Sec. II-C).

A. Planning

The present section focuses on the planning aspect of the framework. Given that there is human supervision throughout the planning process, the proposed approach aims to continuously plan in a finite and local space in a safe and efficient way. Therefore, let us consider an initial point $\mathbf{p}_i \in \mathbb{R}^3$ and a desired point $\mathbf{p}_d \in \mathbb{R}^3$ placed at a constant distance k from \mathbf{p}_i . A path can be defined as a sequence of N points $\mathbf{P} = \{\mathbf{p}_i, \mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_d\} \in \mathbb{R}^3$ with a minimum distance of l m from any obstacle. The method proposed in [6] is implemented as the path planner due to its use of a quadratic program (QP) with the Safe Flight Corridor (SFC) concept. This enable real-time computation of dynamically feasible and safe paths, ensuring the safety and efficiency of the computed paths. It s worth noting that the planning algorithm, was able to run at 100Hz, when executed on a commercial machine equipped with an Intel i7 CPU. As for the mapping aspect, we build upon the work of [11], which introduces the concept of a sliding window approach to mapping. This approach entails continuously updating a local map using spatially sensed data. This method ensures the availability of a reliable map of adequate scope for navigation purposes, all while circumventing the issues of

¹Eurecat, Centre Tecnològic de Catalunya, Robotics and Automation Unit, Cerdanyola del Vallès, Barcelona, Spain. riccardo.franceschini@eurecat.org
julian.cayero@eurecat.org

²Automation and Control group, Department of Electrical Engineering, Danish Technical University, Elektrovej, Building 326, DK-2800 Kgs. Lyngby Denmark mafum@elektro.dtu.dk

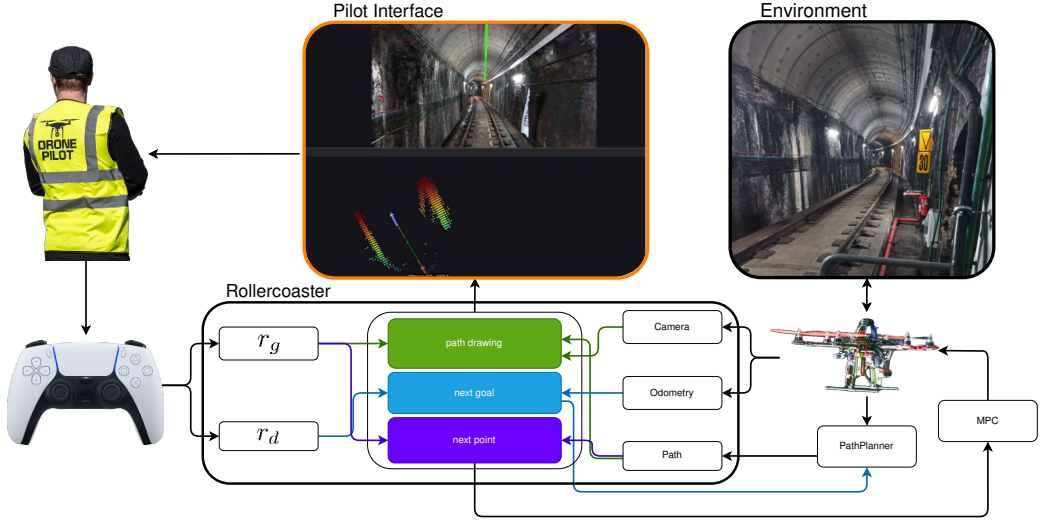


Fig. 1: The provided diagram depicts the interaction between the operator and the UAV. More specifically, the Rollercoaster module receives the gain parameter r_g and direction input r_d from the operator. Simultaneously, it obtains odometry data, camera feedback, and the computed path from the drone. The proposed layer determines the desired next goal by leveraging both the operator's direction input and the drone's odometry information. Subsequently, a path planner generates a safe trajectory towards the intended goal. Following this, utilizing the operator's gain parameter, the desired point and velocity are computed. These values are then communicated to the drone's position controller for execution. Throughout this entire process, the operator remains informed about the progress and trajectory through the overlaid path in the camera feedback, ensuring continuous situational awareness.

drift and the computational demands associated with global mapping methods.

B. Shared Control

To grant operators control over the drone's path and velocity, ensuring obstacle avoidance and safe trajectory planning, a user-friendly interface is provided using a joystick. The operators interact with the joystick and convey two essential inputs to the rollercoaster module: the rollercoaster gain, denoted as $r_g \in [0, 1]$, and the desired direction, represented as $r_d \in [-1, 1]$. The rollercoaster gain r_g control the desired velocity along the path and is used to compute the velocity command as follows. To begin, the path \mathbf{P} from \mathbf{p}_i and \mathbf{p}_0 is discretized into n equally spaced points, and the first point $\mathbf{p}_n \in \mathbf{P}$ is retrieved:

$$p_{n_x} = p_{i_x} + \frac{p_{0_x} - p_{i_x}}{n} \quad (1a)$$

$$p_{n_y} = p_{i_y} + \frac{p_{0_y} - p_{i_y}}{n} \quad (1b)$$

$$p_{n_z} = p_{i_z} + \frac{p_{0_z} - p_{i_z}}{n} \quad (1c)$$

Next, a normalized direction vector \mathbf{d} is obtained by calculating the direction from the current position \mathbf{p}_i to the desired next point \mathbf{p}_n :

$$\mathbf{d} = \frac{\mathbf{p}_n - \mathbf{p}_i}{\|\mathbf{p}_n - \mathbf{p}_i\|} \quad (2)$$

Then, \mathbf{d} is combined with r_g and the maximum drone linear velocity $\mathbf{v}_{max} = \{v_x, v_y, v_z\}$ to compute the desired velocity \mathbf{v}_n :

$$v_{n_x} = d_x r_g v_x \quad (3a)$$

$$v_{n_y} = d_y r_g v_y \quad (3b)$$

$$v_{n_z} = d_z r_g v_z \quad (3c)$$

The desired orientation γ_n needs to point at the desired point \mathbf{p}_n to guarantee a smooth navigation experience, thus is defined as:

$$\gamma_n = \text{atan2}(\mathbf{p}_{i_y} - \mathbf{p}_{n_y}, \mathbf{p}_{i_x} - \mathbf{p}_{n_x}) \quad (4)$$

Finally, the UAV controller receives as position, orientation and velocity, the values \mathbf{p}_n , γ_n , and velocity \mathbf{v}_n . Then, to control the next desired point \mathbf{p}_d the desired direction r_d is used to determine the yaw offset τ :

$$\tau = \arccos(r_d) - \frac{\pi}{2} \quad (5)$$

which is used with current position \mathbf{p}_i , fixed distance k , current orientation γ and angle offset τ , to retrieve the

desired point \mathbf{p}_d :

$$p_{d_x} = p_{i_x} + k \cos(\gamma + \tau) \quad (6a)$$

$$p_{d_y} = p_{i_y} + k \sin(\gamma + \tau) \quad (6b)$$

$$p_{d_z} = p_{i_z} \quad (6c)$$

Algorithm 10 shows the pseudocode for the rollercoaster algorithm.

Algorithm 1 Rollercoaster

```

1: function RC(planner, user_input, drone)
2:    $n \leftarrow 20$   $\triangleright$  Discretize the path into n points
3:   while drone.status is in ROLLERCOASTER do
4:      $\mathbf{p}_i \leftarrow$  drone.pose
5:      $\mathbf{p}_d \leftarrow$  point_ahead( $\mathbf{p}_i, d, user.r_d$ )
6:      $path \leftarrow$  planner.plan( $\mathbf{p}_d$ )
7:      $\mathbf{p}_n \leftarrow$  next_point( $path, n, user.r_g$ )
8:     drone.move_to( $\mathbf{p}_n$ )
9:   end while
10: end function

```

C. Visual Feedback

Maintaining the pilot's involvement and awareness throughout the flight operation is critical. This necessitates the sharing of information and processes occurring within the UAVs with the pilot. To accomplish this, the path obtained through the method outlined in Sec. II-A is consistently overlaid onto the camera stream, offering the operator an immediate view of both the retrieved path, as illustrated in Fig. 4. The camera's field of view is constrained, as indicated

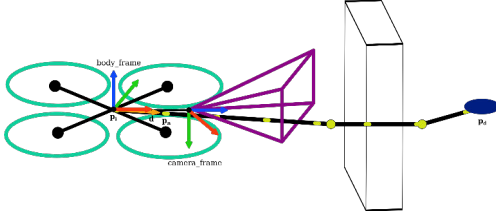


Fig. 2: Drone Frames

in Fig. 2, resulting in not all points having representation within the camera plane. Consequently, alongside the camera feed, a 3D perspective, akin to that depicted in Fig. 3, is concurrently presented to the pilot. This 3D view enables the pilot to gain insight into what the drone perceives and its spatial relationship with surrounding obstacles. Furthermore, considering the operator's control over r_d , as elucidated in the preceding section, the desired points \mathbf{p}_d and \mathbf{p}_n are illustrated as purple and green arrows, respectively. This visual representation provides the pilot with information about the drone's heading and current orientation. Lastly, the operator's control gain r_g is depicted as a green line superimposed over the path, as exemplified in Fig. 4. The length of this line corresponds to the gain value, with

complete overlap when $r_g = 1$ and no line representation when $r_g = 0$. These visual elements collectively empower the pilot to gain insights into the drone's perception and comprehend how the interaction with the UAV influences its behavior. Consequently, this contribute to enhances the operator's overall situational awareness, ultimately resulting in a more enjoyable and secure navigation experience.

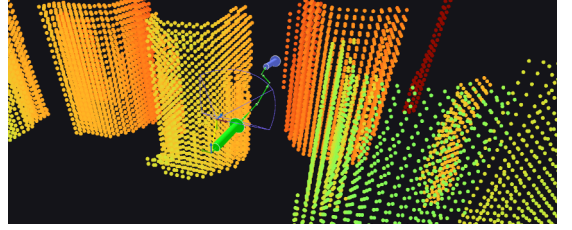


Fig. 3: 3D environment visualization

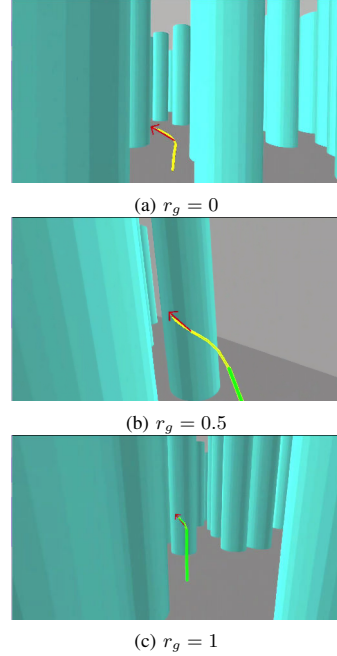


Fig. 4: Visualization of r_g represented as green line over the retrieved path

D. Remote Control

An important aspect of an effective HRI relies on the choice of device used for such interaction. In this case, the proposed approach integrates a PS5 controller due to its commercial availability and the presence of adaptive triggers. As depicted in Figure 5, the operator assumes control over the rollercoaster module, utilizing the left adaptive trigger to

initiate movement and regulate the parameter r_g , while the right thumbstick is employed for adjusting r_d . To enhance the user experience and provide meaningful feedback, the force exerted on the adaptive trigger, denoted as f_t , is dynamically adjusted based on the value of r_g . Specifically, the relationship is defined as follows:

$$f_t = \min((255 \cdot r_g) \cdot \lambda, 255) \quad (7)$$

This equation ensures that the adaptive trigger force, represented on a scale of 0 to 255, is scaled proportionally with r_g and then further amplified by a factor of $\lambda = 1.2$ manually chosen. Doing so, the operator receives a stronger feedback response at lower velocities.



Fig. 5: Remote control interaction

III. CONCLUSION

In conclusion, the Rollercoaster framework presents an innovative approach to enhance the human-robot interaction (HRI) in teleoperating unmanned aerial vehicles. The operator actively participates in the planning process and retains control over the UAV's velocity using the r_g parameter and direction with r_d . This collaborative approach empowers the operator to guide the drone's movement along the intended path, while the drone autonomously ensures safety and interprets spatial data. By incorporating extended reality (XR) elements such as the overlaid trajectory and haptic feedback through the adaptive triggers of the PS5 controller, this framework improves the operator's situational awareness while reducing the mental effort required for drone piloting. Moving forward, there will be a focus on user validation to better assess the effectiveness of the approach. Subsequent research efforts will prioritize the exploration of real-world applications and the advancement of the integration of emerging technologies, including immersive displays and haptic devices, to further enhance HRI experiences.

ACKNOWLEDGMENT

This work has been supported by the European Unions Horizon 2020 Research and Innovation Programme AERO-TRAIN under Grant Agreement No. 953454.

REFERENCES

- [1] Jacopo Aleotti et al. "Detection of Nuclear Sources by UAV Teleoperation Using a Visuo-Haptic Augmented Reality Interface". In: *Sensors* 17.10 (2017). ISSN: 1424-8220. DOI: 10.3390/s17102234. URL: <https://www.mdpi.com/1424-8220/17/10/2234>.
- [2] Sony Interactive Entertainment. *DualSense Wireless Controller*. <https://www.playstation.com/en-us/accessories/dualsense-wireless-controller/>. 2021.
- [3] Riccardo Franceschini, Matteo Fumagalli, and Julian Cayero Becerra. "Enhancing Human-Drone Interaction with Human-Meaningful Visual Feedback and Shared-Control Strategies". In: *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2023, pp. 1162–1167. DOI: 10.1109/ICUAS57906.2023.10156190.
- [4] Tsung-Chi Lin, Achyuthan Unni Krishnan, and Zhi Li. "Comparison of Haptic and Augmented Reality Visual Cues for Assisting Tele-manipulation". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 9309–9316. DOI: 10.1109/ICRA46639.2022.9811669.
- [5] Chuhao Liu and Shaojie Shen. "An Augmented Reality Interaction Interface for Autonomous Drone". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 11419–11424. DOI: 10.1109/IROS45743.2020.9341037.
- [6] Sikang Liu et al. "Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments". In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1688–1695. DOI: 10.1109/LRA.2017.2663526.
- [7] Dylan P. Losey and Marcia K. O'Malley. "Trajectory Deformations From Physical Human-Robot Interaction". In: *IEEE Transactions on Robotics* 34.1 (2018), pp. 126–138. DOI: 10.1109/TRO.2017.2765335.
- [8] Carlo Masone et al. "Semi-autonomous trajectory generation for mobile robots with integral haptic shared control". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 6468–6475. DOI: 10.1109/ICRA.2014.6907814.
- [9] F.J. Perez-Grau et al. "Semi-autonomous teleoperation of UAVs in search and rescue scenarios". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2017, pp. 1066–1074. DOI: 10.1109/ICUAS.2017.7991349.
- [10] P. Ramon-Soria et al. "Planning System for Integrated Autonomous Infrastructure Inspection using UAVs". In: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2019, pp. 313–320. DOI: 10.1109/ICUAS.2019.8797874.

- [11] Jesus Tordesillas et al. "Real-Time Planning with Multi-Fidelity Models for Agile Flights in Unknown Environments". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 725–731. DOI: 10.1109/ICRA.2019.8794248.
- [12] Karl D. von Ellenrieder et al. "Shared human-robot path following control of an unmanned ground vehicle". In: *Mechatronics* 83 (2022), p. 102750. ISSN: 0957-4158. DOI: <https://doi.org/10.1016/j.mechatronics.2022.102750>. URL: <https://www.sciencedirect.com/science/article/pii/S0957415822000083>.
- [13] Michael E. Walker, Hooman Hedayati, and Daniel Szafrir. "Robot Teleoperation with Augmented Reality Virtual Surrogates". In: *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2019, pp. 202–210. DOI: 10.1109/HRI.2019.8673306.
- [14] Mohammad Kassem Zein et al. "Deep Learning and Mixed Reality to Autocomplete Teleoperation". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 4523–4529. DOI: 10.1109/ICRA48506.2021.9560887.
- [15] Mohammad Kassem Zein et al. "Enhanced Teleoperation Using Autocomplete". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9178–9184. DOI: 10.1109/ICRA40945.2020.9197140.

Riding the Rollercoaster: Improving UAV Piloting Skills with Augmented Visualization and Collaborative Planning

Riccardo Franceschini^{1,2}, Javier Rodriguez Marquez¹, Matteo Fumagalli² and Julian Cayero Becerra¹

Abstract—Operating unmanned aerial vehicles (UAVs) in complex environments can be challenging, particularly for inexperienced operators. This paper introduces a method aimed at enhancing the piloting experience by incorporating an intermediary processing layer between the remote controller and the drone. The presented approach empowers operators to control both the speed and direction of the UAV along a secure path, which is continuously computed and overlaid onto the operator's camera stream. The UAV autonomously plans and executes this path, adapting to the operator's commands and environmental changes. The primary aim of this proposed solution is to enhance the operator's situational awareness, perception, as well as the safety and efficiency of UAV navigation. The paper outlines the system and methodology employed, showing its ability to operate at a high enough frequency to enable seamless user interactions. Furthermore, to validate the effectiveness of this approach, a real-world test and a user-based experimental study conducted in a simulation environment with an audience comprising varying levels of pilot expertise have been carried out.

I. INTRODUCTION

Teleoperation remains a predominant mode of human-robot interaction in contemporary robotic systems, notably for unmanned aerial vehicles (UAVs). Collaborative efforts between drones and operators are essential for ensuring safety in tasks demanding precise navigation within cluttered and hazardous environments, as highlighted in [1, 19]. Manual teleoperation currently serves as the de facto standard. However, navigating a drone in such scenarios poses significant challenges for pilots, particularly novice users, who may experience restricted situational awareness and limited visual feedback when the UAVs are at a distance. To enhance user experience, the literature has proposed various methods to promote collaboration between operators and robots, achieved through extended reality (XR) interfaces and shared control strategies [21], which are further elaborated in Sec.II. This work propose a system that aims to combine augmented user interfaces and shared control strategies to address the challenge of safe and effective navigation. The system relies on an efficient path planning pipeline that continuously computes safe routes for the drone, while the operator is responsible for defining the UAV's direction and speed along these routes. Supported by the findings of [2], which demonstrated the effectiveness of augmented visualization in user

acceptance, the interaction is facilitated through an enhanced visualization pipeline that displays relevant information on the user interface. The operator uses a SONY DualSense™ Wireless Controller [5] to specify the direction and velocity of the UAV. The controller also provides haptic feedback to the operator, giving continuous feedback to the user. This interaction framework, called Rollercoaster, aims to increase the operator's situational awareness and facilitate the UAV piloting experience. Hence, the objective of this study is to demonstrate the interaction between the operator and the UAV and to substantiate its effectiveness. Experiments were conducted in a simulated environment where crashes were possible, involving pilots with varying levels of expertise, to validate the efficacy of the interaction. These experiments had the objective of evaluating their performance and gathering user subjective feedback while they navigated through a randomly generated map, both with and without the proposed system's implementation. The paper's structure is as follows: At first a literature overview of is proposed in Sec.II, then in Sec.III, the processing layer of the proposed system is outlined, addressing the collaborative path planning procedure and augmented visualization. Subsequently, in Sec.IV, an analysis of the collected data is presented and an evaluation over a real world experiment is conducted in Sec.V. Finally, in Sec.VI, the conclusion and future directions of the research are outlined.

II. RELATED WORK

The landscape of human-robot interaction, particularly in teleoperation scenarios, has witnessed diverse approaches to enhance user experience and system performance. Existing works have explored various strategies, encompassing extended reality (XR) interfaces, shared control mechanisms, haptic feedback, deep learning integration, path planning, and collision avoidance. Specifically, regarding drone control, different approaches have been attempted, falling into macro categories.

Assisted Pilot Interfaces: The first category involves the development of systems such as [18, 17], where the operator controls the UAV velocity in a manner similar to normal teleoperation. A collision avoidance algorithm is deployed between the operator and UAV to adjust the operator's commands when they pose a collision risk.

Augmented Reality Interfaces for Point and Move: Works from [10, 24] showcase interfaces for controlling almost fully autonomous drones. These approaches require the use of head-mounted displays (HMD), and their interaction revolves around visualizing 3D data collected by the drone or

¹Eurecat, Centre Tecnològic de Catalunya, Robotics and Automation Unit, Cerdanyola del Vallès, Barcelona, Spain. [riccardo.franceschini, julian.cayero, javier.rodriguez]@eurecat.org

²Automation and Control group, Department of Electrical Engineering, Danish Technical University, Elektrovej, Building 326, DK-2800 Kgs. Lyngby Denmark mafum@elektro.dtu.dk

visualizing the drone's trajectory superimposed on a physical drone in front of the operator. The primary interaction here is centered on defining a point and controlling the UAV movement with the proposed interfaces.

Haptic Control for Trajectory Manipulation: Other approaches, such as [3, 13, 12, 23, 14], revolve around the use of haptic devices for trajectory modification, enhancing the control experience by proposing different haptic feedback schemes to the user. These methods aim to include the operator in the UAV control pipeline, allowing the operator to perform vertical and lateral displacement over a predefined trajectory, providing haptic feedback based on the feasibility of the movement from an obstacle and platform perspective.

Deep Learning Teleoperation: Other compelling teleoperation approaches are proposed by [27, 26], where the authors suggest learning common trajectories, such as turns, straight lines, or circle trajectory shapes, using a deep learning agent that reads the operator's control input. The agent is then deployed between the operator and the UAV and takes control when it detects that the operator is attempting one of those trajectories, executing a precise movement.

Gaze Control Navigation: Interesting interaction approaches are also proposed by [25, 6], where the operator's gaze is detected through an HMD and used either to understand the operator's intention, correct the UAV trajectory, or define a point where the UAV should move.

Rollercoaster: Our approach introduces a navigation interaction akin to classical teleoperation schemes discussed in the Deep Learning and Assisted pilot methods, aiming to maintain a familiar interaction scheme easily integrable with manual pilot teleoperation. However, instead of depending on collision avoidance systems or deep learning agents, we deploy a fast planner between the operator and the UAV. This planner continuously seeks collision-free paths, and its activation or deactivation is simplified with the press of a button, granting complete control to the operator. This setup is complemented by an immersive interface displaying the retrieved plan and additional information in real-time on the remote display.

III. METHODOLOGY

This section explains the methodology employed for planning and distributing control in the bilateral control execution. The approach is sensor-agnostic, but it presupposes that the UAV has the capability to estimate its odometry and perceive its surrounding 3D environment, whether through depth cameras or LiDAR. Initially, this section outlines the data planning and mapping process (Sec. III-A). Following that, it introduces an architecture designed for the integration of the operator into the planning loop (Sec. III-B). Lastly, the visualization technique is detailed (Sec. III-C).

A. Planning

The present section focuses on the planning aspect of the framework. Given that there is human supervision throughout the planning process, the proposed approach aims to continuously plan in a finite and local space in a safe and efficient

way. Therefore, let us consider an initial point $\mathbf{p}_i \in \mathbb{R}^3$ and a desired point $\mathbf{p}_d \in \mathbb{R}^3$ placed at a constant distance k from \mathbf{p}_i . A path can be defined as a sequence of N points $\mathbf{P} = \{\mathbf{p}_i, \mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_d\}$ with a minimum distance of l m from any obstacle. The method introduced in [11] has been implemented as the path planner due to its utilization of a quadratic program (QP) featuring the Safe Flight Corridor (SFC) concept. The SFC consists of a collection of convex, overlapping polyhedra that models the available free space and establishes a continuous path from the robot's current position to the goal destination. Through the SFC, a set of linear inequality constraints is introduced into the QP, allowing for real-time motion planning. This capability enables the real-time computation of dynamically feasible and secure pathways, thus ensuring both safety and efficiency in path calculation. Notably, the planning algorithm was able to achieve a running frequency of 100Hz when executed on a commercial machine equipped with an Intel i7 CPU. Regarding the mapping aspect, the methodology builds upon the research presented in [22]. This work introduces the concept of a sliding window approach to mapping, characterized by the continuous updating of a local map using spatially sensed data. This technique guarantees the availability of a dependable map with sufficient coverage for navigation requirements, while simultaneously mitigating the challenges related to drift and the computational resources demanded by global mapping methods.

B. Shared Control

To grant operators control over the drone's path and velocity, ensuring obstacle avoidance and safe trajectory planning, a user-friendly interface is provided using a joystick. The operators interact with the joystick and convey two essential inputs to the Rollercoaster module: the rollercoaster gain, denoted as $r_g \in [0, 1]$, and the desired direction, represented as $r_d \in [-1, 1]$. The rollercoaster gain r_g control the desired velocity along the path and is used to compute the velocity command as follows. To begin, the path \mathbf{P} from \mathbf{p}_i to \mathbf{p}_0 is discretized into n equally spaced points, and the first point $\mathbf{p}_n \in \mathbf{P}$ is retrieved:

$$\mathbf{p}_n = \mathbf{p}_i + \frac{\mathbf{p}_0 - \mathbf{p}_i}{n} \quad (1)$$

Next, a normalized direction vector \mathbf{d} is obtained by calculating the direction from the current position \mathbf{p}_i to the desired next point \mathbf{p}_n :

$$\mathbf{d} = \frac{\mathbf{p}_n - \mathbf{p}_i}{\|\mathbf{p}_n - \mathbf{p}_i\|} \quad (2)$$

Then, \mathbf{d} is combined with r_g and the maximum drone linear velocity $\mathbf{v}_{max} = \{v_x, v_y, v_z\}$ to compute the desired velocity \mathbf{v}_n :

$$v_{n_x} = d_x r_g v_x \quad (3a)$$

$$v_{n_y} = d_y r_g v_y \quad (3b)$$

$$v_{n_z} = d_z r_g v_z \quad (3c)$$

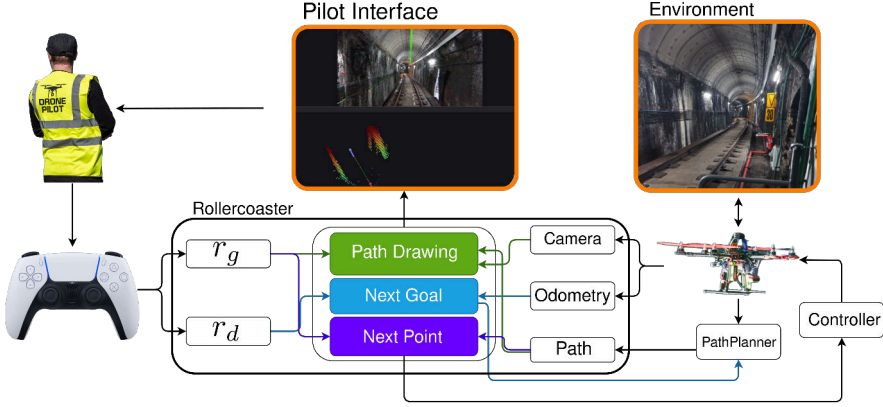


Fig. 1: Diagram depicting the interaction between the operator and the UAV. More specifically, the Rollercoaster module receives the gain parameter r_g and direction input r_d from the operator. Simultaneously, it obtains odometry data, camera feedback, and the computed path from the drone. The proposed layer determines the desired next goal by leveraging both the operator's direction input and the drone's odometry information. Subsequently, a path planner generates a safe trajectory towards the intended goal. Following this, utilizing the operator's gain parameter, the desired point and velocity are computed. These values are then communicated to the drone's inner controller for execution. Throughout this entire process, the operator remains informed about the progress and trajectory through the overlaid path in the camera feedback, ensuring continuous situational awareness

The desired orientation γ_n needs to point at the desired point \mathbf{p}_n to guarantee a smooth navigation experience, thus is defined as:

$$\gamma_n = \text{atan2}(\mathbf{p}_{iy} - \mathbf{p}_{ny}, \mathbf{p}_{ix} - \mathbf{p}_{nx}) \quad (4)$$

Finally, the UAV controller receives as position, orientation and velocity, the values \mathbf{p}_n , γ_n , and velocity \mathbf{v}_n . Then, to control the next desired point \mathbf{p}_d the desired direction r_d is used to determine the yaw offset τ :

$$\tau = \arccos(r_d) \quad (5)$$

which is used with current position \mathbf{p}_i , fixed distance k , current orientation γ and angle offset τ , to retrieve a new desired point \mathbf{p}_d :

$$p_{dx} = p_{ix} + k \cos(\gamma + \tau) \quad (6a)$$

$$p_{dy} = p_{iy} + k \sin(\gamma + \tau) \quad (6b)$$

$$p_{dz} = p_{iz} \quad (6c)$$

To point out is that the path is discretized, and only the initial position is sent to the controller, along with the velocity. This approach minimizes discrepancies in UAV movement, which is crucial given the high-frequency repetition of this process. Larger displacements have the potential to impact the behavior of the position controller. Algorithm 10 shows the pseudocode for the rollercoaster algorithm.

C. Visual Feedback

Maintaining the pilot's involvement and awareness throughout the flight operation is critical. This necessitates the sharing of information and processes occurring within the

Algorithm 1 Rollercoaster

```

1: function RC(planner, user_input, drone)
2:    $n \leftarrow 20$   $\triangleright$  Discretize the path into n points
3:   while drone.status is in ROLLERCOASTER do
4:      $\mathbf{p}_i \leftarrow \text{drone.pose}$ 
5:      $\mathbf{p}_d \leftarrow \text{point\_ahead}(\mathbf{p}_i, \mathbf{d}, \text{user}.r_d)$ 
6:      $\text{path} \leftarrow \text{planner.plan}(\mathbf{p}_d)$ 
7:      $\mathbf{p}_n \leftarrow \text{next\_point}(\text{path}, n, \text{user}.r_g)$ 
8:     drone.move.to( $\mathbf{p}_n$ )
9:   end while
10: end function

```

UAVs with the pilot. To accomplish this, the path obtained through the method outlined in Sec. III-A is consistently overlaid onto the camera stream, offering the operator an immediate view of the retrieved path, as illustrated in Figure 4. The camera's field of view is constrained, as indicated

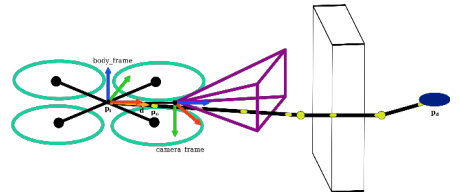


Fig. 2: Representation of drones frames, retrieved path and camera FOV (Field of View)

in Figure 2, resulting in not all points having representation within the camera plane. Consequently, alongside the camera feed, a 3D perspective, akin to that depicted in Figure 3, is concurrently presented to the pilot. This 3D view enables the pilot to gain insight into what the drone perceives and its spatial relationship with surrounding obstacles. Furthermore, considering the operator's control over r_d , as elucidated in the preceding section, the desired points \mathbf{p}_d and \mathbf{p}_n are illustrated as purple and green arrows, respectively. This visual representation provides the pilot with information about the drone's heading and current orientation. Lastly, the operator's control gain r_g is depicted as a green line superimposed over the path, as exemplified in Figure 4. The length of this line corresponds to the gain value, with complete overlap when $r_g = 1$ and no line representation when $r_g = 0$. These visual elements collectively empower the pilot to gain insights into the drone's perception and comprehend how the interaction with the UAV influences its behavior. Consequently, this contributes to enhance the operator's overall situational awareness, ultimately resulting in a more enjoyable and secure navigation experience.

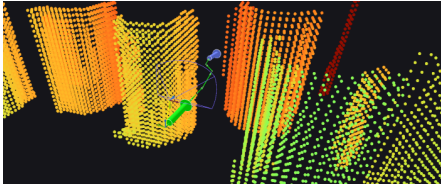


Fig. 3: 3D environment visualization.

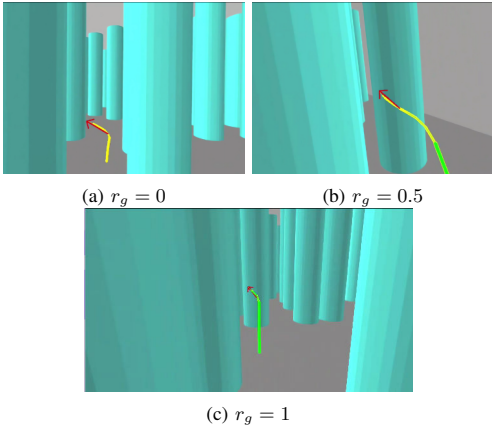


Fig. 4: Visualization of r_g at different values represented as a green line over the retrieved path depicted in yellow

D. Remote Control

An important aspect of an effective HRI relies on the choice of device used for such interaction. In this case,

the proposed approach integrates a PS5 controller due to its commercial availability and the presence of adaptive triggers. As illustrated in Figure 5, the operator takes charge of the Rollercoaster module, using the left adaptive trigger to initiate movement and adjust the parameter r_g , while the right thumbstick controls r_d through lateral displacement. To enhance the user experience and provide meaningful feedback, the force exerted on the adaptive trigger, denoted as f_t , is dynamically adjusted based on the value of r_g . Specifically, the relationship is defined as follows:

$$f_t = \min((255 \cdot r_g) \cdot \lambda, 255) \quad (7)$$

This equation ensures that the adaptive trigger force, represented on a scale of 0 to 255, is scaled proportionally with r_g and then further amplified by a factor of $\lambda = 1.2$ chosen for convenience. This amplification ensures that the operator receives a stronger feedback response at lower velocities.

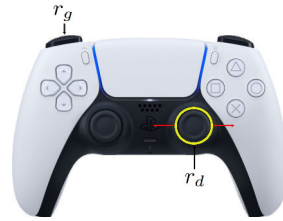


Fig. 5: Remote control interaction

IV. EXPERIMENT

To validate the proposed approach, a study for evaluation is put forth. The study's scope encompasses both empirical and subjective assessment through utilization of the NASA-TLX framework [7]. The study aims to ascertain whether the suggested approach enhances the navigation capabilities and experiences of pilots with different level of expertise.

A. Experiment Setting

Thus, 35 participants were recruited from the research institution and through word of mouth. The participants' ages ranged between 24 and 61 years old (Mean = 33.61, SD = 9.33). They were requested to rate their UAV piloting experience on a scale of 1 to 5, where 1 indicated no experience and 5 denoted proficiency (Mean = 1.77, SD = 1.19). The experiment is performed within a simulated environment utilizing ROS [20], Gazebo [9], and PX4-SITL [15] with the cascade PID as the UAV's controller (Figure. 9 and 8). A map measuring 100x100 meters was generated in Blender [4], placing 550 randomly positioned pipe-like vertical obstacles. This configuration resulted in a densely cluttered and intricate environment, as illustrated in Figure 6. Participants were tasked with navigating toward a designated goal, depicted as a green pipe (Figure 7). To enhance orientation within the maze, an orange dot (Figure 7) was overlaid on the camera feedback. This dot represents

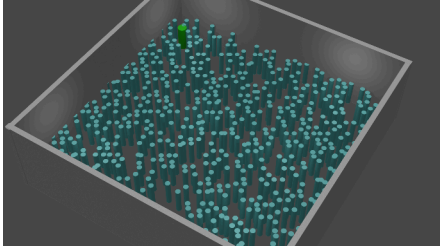


Fig. 6: Experiment environment

the direction of the goal, facilitating the navigation process. Following the completion of an initial training phase, where

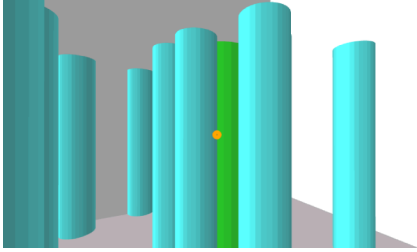


Fig. 7: Camera feedback during the experiments, goal is highlighted as a green pipe, and goal direction is given by the orange dot

the user acquired proficiency for 5 minutes in piloting the UAV both with and without the Rollercoaster, the actual testing phase was initiated. This phase involved attempting to solve the maze three times without the Rollercoaster and three more times with the option of using it. The experiment failed if the simulation detected a collision. Participants could use the Rollercoaster as they wished by controlling r_g , but it was not mandatory. This approach aimed to capture the most natural interaction with the system and to understand how users interacted with it. Throughout the experiment, we monitored the drone's position, orientation, and user interactions, including r_g and r_d . Time tracking was implemented during the evaluation but without imposing constraints on users, as the map complexity was already high. We chose not to report time results due to insignificant differences in average times for successful experiments.

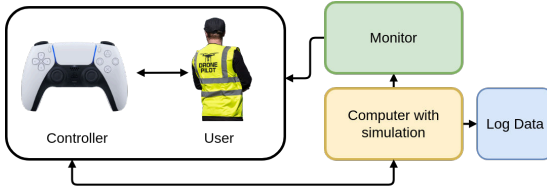


Fig. 8: Experiment Schema

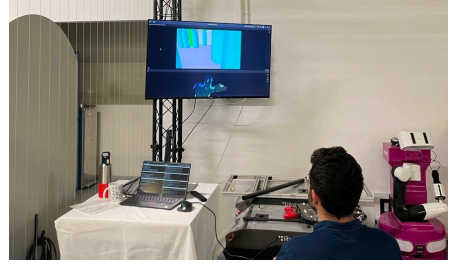


Fig. 9: Laboratory experimental setup

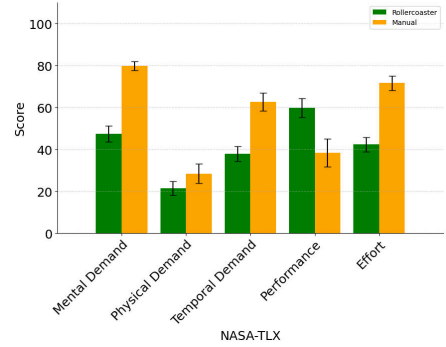


Fig. 10: Nasa-TLX results. Error bars are standard error

B. Experiment Result

The test results are then being analyzed. At first, the survey outcomes (Figure 10) is considered. The data highlights a meaningful distinction between the two approaches. This difference translates to a piloting experience that is less stressful in terms of mental and physical demands, making it easier with reduced effort while at the same time increasing the performance. It is noteworthy that the only parameters that exhibit substantial similarity between the two modes pertain to the level of physical intensity. Despite the slight difference, it is conceivable that this resemblance in intensity stems from the necessity of coordinating actions to maintain pressure on the velocity trigger while simultaneously focusing on maintaining the desired direction in the Rollercoaster mode. This convergence of physical demands results in a similarity between controlling the UAV manually and using the Rollercoaster. The trend revealed in the survey, indicating an improvement in the piloting experience, is also clearly reflected in the success rate (Figure 11). While the overall success rate remains relatively low due to the intricacy of the map, the completion rate with the Rollercoaster is approximately three times higher compared to when it's not used. This finding further reinforces the validity of the previously demonstrated outcome.

It is also noteworthy to highlight the disparity in success rates when taking participants' experience levels into account, as illustrated in Fig. 12 (with none of the participants

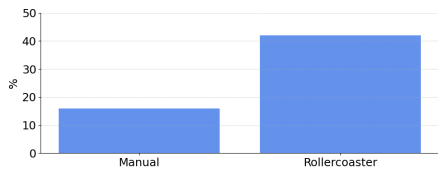


Fig. 11: Task completion Success Rate

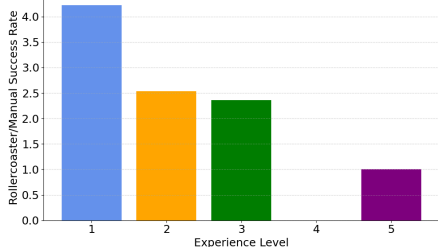


Fig. 12: Success rate improvement at different experience level

rating themselves as experience level 4). Interestingly, there is a substantial improvement in performance, particularly evident when a pilot lacks confidence in their flying skills (up to a four-fold improvement for non-expert pilots). Conversely, the performance gain is either diminished or negligible when a pilot is considered an expert. This phenomenon is believed to be a result of the user's confidence in their own abilities, which, in turn, leads to skepticism regarding the system and, consequently, difficulty in fully harnessing its advantages. Another interesting point that supports the effectiveness of the Rollercoaster is how it was used. Since participants were encouraged but not forced to use it, its usage varied during each run. When looking at the success rates (Figure 13), it's interesting to notice that the runs where pilots used the Rollercoaster more confidently and for a higher percentage of the time with the proposed solution were also the ones where they were more likely to succeed.

Furthermore, when unsuccessful runs are taken into account, the trend persists. Rollercoaster trials exhibit a tendency to approach, on average, a closer distance to the goal compared to the manual attempts (Figure 14). This reaffirms the Rollercoaster's efficacy in enhancing the pilot's experience while navigating through complex environments.

Therefore, the results highlighted a significant overall performance improvement achieved by users, with a noteworthy correlation between Rollercoaster usage and expertise. This demonstrates the framework's capability to enhance the experience, especially for novice users. The findings also indicated that, on average, a higher level of system usage positively correlated with mission success. Furthermore, even in instances where the task was not completed, participants were able to approach the goal more closely on average. However, as the primary goal of the system is to enhance the

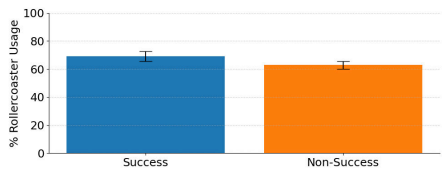


Fig. 13: Rollercoaster usage. Error bars are standard error

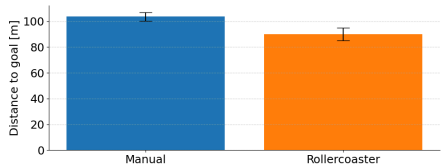


Fig. 14: Distance to the goal on unsuccessful trials. Error bars are standard error

piloting experience while maintaining the decision-making process, it's important to acknowledge the possibility of crashes. During the experiment, several noteworthy patterns leading to crashes emerged. Occasionally, participants exhibited a lack of confidence in the system, leading them to either release the trigger from the Rollercoaster mode while was in a dangerous situation or make rapid directional changes to the planner. These behaviors were more pronounced when the UAV had to navigate through tight column spaces. This raised concerns about the planner's execution behavior, prompting participants to switch between the Rollercoaster and manual modes, inadvertently causing unintended crashes. Experience often helps mitigate such behaviors, yet they could also be addressed through improved interaction design. For instance, the direct control of the UAV's r_d using the stick might be replaced by a smoother transition, thus potentially reducing the likelihood of such occurrences.

V. REAL-WORLD TESTING

Despite the proposed work's primary focus on evaluating the interaction framework, a test was also conducted to demonstrate the framework's real capability. This test was performed on a FPV-like drone equipped with a Vox12[16] which as integrated PX4[15] and a RealSense d435[8] serving as the primary depth sensor. The test involved navigating toward an obstacle, as depicted in the setup Figure 15.

Figure 16 illustrates the disparities between the user input and the actual trajectory followed by the UAV. In this scenario, the operator only controlled the drone's velocity and direction, while the UAV itself was responsible for navigating around obstacles. The arrows in the plot represent the r_d , which on purpose were pointed towards the obstacle. However, as indicated by the dots representing the UAV trajectory, the obstacle was successfully avoided.

While performing the navigation, the operator remained



Fig. 15: Test setup

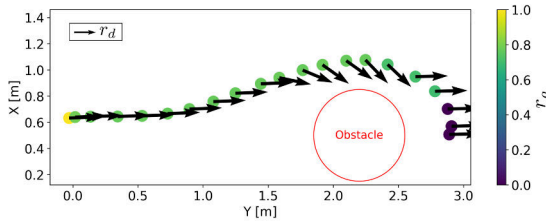


Fig. 16: UAV trajectory

aware of the ongoing process through the augmented interface, as shown in Figure 17.

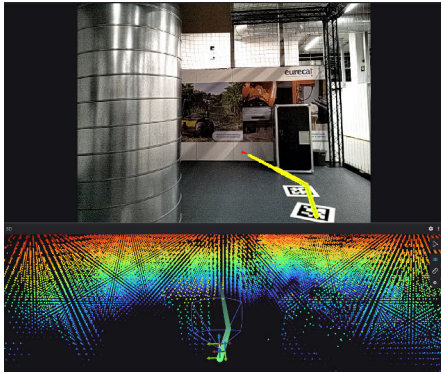


Fig. 17: Screenshot of the pilot UI view during flight

VI. CONCLUSION

In conclusion, the Rollercoaster framework presents an innovative approach to enhance the human-robot interaction (HRI) in teleoperating unmanned aerial vehicles. The operator actively participates in the planning process and retains control over the UAV's velocity using the r_g parameter and direction with r_d . This collaborative approach empowers the operator to guide the drone's movement along the intended path, while the drone autonomously ensures safety and interprets spatial data. Through user testing, this framework has been demonstrated to enhance the operator's situational

awareness and reduce the mental effort required for drone piloting by improving the interaction experience with elements such as the overlaid trajectory and haptic feedback via the adaptive triggers of the PS5 controller. Additionally, it is crucial to emphasize the performance gains observed when using the Rollercoaster system with a non-expert audience, highlighting its advantages, especially for novice pilots. Moving forward, future research will concentrate on refining interaction design to address the aforementioned issues, taking into account the integration of emerging technologies such as immersive displays and haptic devices to enhance human-robot interaction experiences. Finally, further real-world tests will be conducted to more thoroughly assess how communication delays, state estimation drift, and sensor noise could impact the overall experience.

ACKNOWLEDGMENT

This work has been supported by the European Unions Horizon 2020 Research and Innovation Programme AERO-TRAIN under Grant Agreement No. 953454.

REFERENCES

- [1] Jacopo Aleotti et al. "Detection of Nuclear Sources by UAV Teleoperation Using a Visuo-Haptic Augmented Reality Interface". In: *Sensors* 17.10 (2017). ISSN: 1424-8220. DOI: 10.3390/s17102234. URL: <https://www.mdpi.com/1424-8220/17/10/2234>.
- [2] Connor Brooks and Daniel Szafrir. "Visualization of Intended Assistance for Acceptance of Shared Control". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 11425–11430. DOI: 10.1109/IROS45743.2020.9340964.
- [3] Jonathan Cacace, Alberto Finzi, and Vincenzo Lippiello. "A mixed-initiative control system for an Aerial Service Vehicle supported by force feedback". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 1230–1235. DOI: 10.1109/IROS.2014.6942714.
- [4] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: <http://www.blender.org>.
- [5] Sony Interactive Entertainment. *DualSense Wireless Controller*. <https://www.playstation.com/en-us/accessories/dualsense-wireless-controller/>. 2021.
- [6] Okan Erat et al. "Drone-Augmented Human Vision: Exocentric Control for Drones Exploring Hidden Areas". In: *IEEE Transactions on Visualization and Computer Graphics* 24.4 (2018), pp. 1437–1446. DOI: 10.1109/TVCG.2018.2794058.
- [7] Sandra G Hart and Lowell E Staveland. "Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research". In: *Advances in psychology* 52 (1988), pp. 139–183.

- [8] Leonid Keselman et al. "Intel RealSense Stereoscopic Depth Cameras". In: *CoRR* abs/1705.05548 (2017). arXiv: 1705.05548. URL: <http://arxiv.org/abs/1705.05548>.
- [9] N. Koenig and A. Howard. "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.
- [10] Chuhao Liu and Shaojie Shen. "An Augmented Reality Interaction Interface for Autonomous Drone". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 11419–11424. DOI: 10.1109/IROS45743.2020.9341037.
- [11] Sikang Liu et al. "Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments". In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1688–1695. DOI: 10.1109/LRA.2017.2663526.
- [12] Dylan P. Losey and Marcia K. O'Malley. "Trajectory Deformations From Physical Human-Robot Interaction". In: *IEEE Transactions on Robotics* 34.1 (2018), pp. 126–138. DOI: 10.1109/TRO.2017.2765335.
- [13] Carlo Masone et al. "Semi-autonomous trajectory generation for mobile robots with integral haptic shared control". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 6468–6475. DOI: 10.1109/ICRA.2014.6907814.
- [14] Carlo Masone et al. "Shared planning and control for mobile robots with integral haptic feedback". In: *The International Journal of Robotics Research* 37.11 (2018), pp. 1395–1420. DOI: 10.1177/0278364918802006. eprint: <https://doi.org/10.1177/0278364918802006>. URL: <https://doi.org/10.1177/0278364918802006>.
- [15] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 6235–6240. DOI: 10.1109/ICRA.2015.7140074.
- [16] ModalAI. *VOXL2: A Powerful Companion Computer for Drones*. <https://www.modalai.com/products/voxl-2?variant=39914779836467>. Accessed 2024-02-08.
- [17] Marcin Odelga, Paolo Stegagno, and Heinrich H. Bühlhoff. "Obstacle detection, tracking and avoidance for a teleoperated UAV". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 2984–2990. DOI: 10.1109/ICRA.2016.7487464.
- [18] Marcin Odelga et al. "A Self-contained Teleoperated Quadrotor: On-Board State-Estimation and Indoor Obstacle Avoidance". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 7840–7847. DOI: 10.1109/ICRA.2018.8463185.
- [19] F.J. Perez-Grau et al. "Semi-autonomous teleoperation of UAVs in search and rescue scenarios". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2017, pp. 1066–1074. DOI: 10.1109/ICUAS.2017.7991349.
- [20] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. Version ROS Melodic Morenia. May 23, 2018. URL: <https://www.ros.org>.
- [21] Ryo Suzuki et al. "Augmented Reality and Robotics: A Survey and Taxonomy for AR-Enhanced Human-Robot Interaction and Robotic Interfaces". In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI '22. New Orleans, LA, USA: Association for Computing Machinery, 2022. ISBN: 9781450391573. DOI: 10.1145/3491102.3517719. URL: <https://doi.org/10.1145/3491102.3517719>.
- [22] Jesus Tordesillas et al. "Real-Time Planning with Multi-Fidelity Models for Agile Flights in Unknown Environments". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 725–731. DOI: 10.1109/ICRA.2019.8794248.
- [23] Karl D. von Ellenrieder et al. "Shared human-robot path following control of an unmanned ground vehicle". In: *Mechatronics* 83 (2022), p. 102750. ISSN: 0957-4158. DOI: <https://doi.org/10.1016/j.mechatronics.2022.102750>. URL: <https://www.sciencedirect.com/science/article/pii/S0957415822000083>.
- [24] Michael E. Walker, Hooman Hedayati, and Daniel Szafir. "Robot Teleoperation with Augmented Reality Virtual Surrogates". In: *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2019, pp. 202–210. DOI: 10.1109/HRI.2019.8673306.
- [25] Qianhao Wang et al. "GPA-Teleoperation: Gaze Enhanced Perception-Aware Safe Assistive Aerial Teleoperation". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5631–5638. DOI: 10.1109/LRA.2022.3153898.
- [26] Mohammad Kassem Zein et al. "Deep Learning and Mixed Reality to Autocomplete Teleoperation". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 4523–4529. DOI: 10.1109/ICRA48506.2021.9560887.
- [27] Mohammad Kassem Zein et al. "Enhanced Teleoperation Using Autocomplete". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9178–9184. DOI: 10.1109/ICRA40945.2020.9197140.

Point, Segment, and Inspect: Leveraging Promptable Segmentation Models for Semi-Autonomous Aerial Inspection

Riccardo Franceschini^{1,2}, Javier Rodriguez Marquez¹, Matteo Fumagalli² and Julian Cayero Becerra¹

Abstract—Operating unmanned aerial vehicles (UAVs) for assets inspections poses distinct challenges encompassing the need to maintain a safe distance from the inspection area, ensure correct orientation towards the inspected surface, and achieve comprehensive coverage of the entire surface. Achieving these tasks is inherently complex and stressful. Therefore, a novel approach that seeks to enhance the piloting experience by harnessing the latest advancements in segmentation models, such as Segment Anything Model (SAM), is proposed. These models, thanks to their prompting capabilities, allow seamless communication between the operator and the UAV, opening up the possibility of defining intricate inspection regions through simple interactions. Within this approach, decision-making authority remains with the operator, while the UAV takes on the demanding task of segmenting the designated area and devising an appropriate traversal plan. Throughout this process, the operator’s situational awareness is heightened through visual cues overlaid on the camera stream and a 3D panel presenting information of the drone position and spatially sensed data. This teleoperation framework allows the operator to maintain continuous control of the ongoing operation through a simplified interface. The paper delineates both the system and the methodology employed, showcasing the effectiveness of integrating segmentation models into the decision-making workflow. The validity of the proposed framework is established through testing within a photorealistic UAV simulator along with real experiments in a controlled laboratory environment.

I. INTRODUCTION

Teleoperation remains a prevalent approach for human-robot interaction in modern robotics, especially with unmanned aerial vehicles (UAVs). The collaboration between the drones and the operators is crucial in ensuring safety in tasks that require precise navigation in difficult to navigate and critical environments, as mentioned in [1, 25]. To enhance user experience, the literature has proposed diverse methods to foster teamwork between operators and robots, achieved through extended reality (XR) interfaces and shared control strategies [29]. For instance, in [21, 7], XR was utilized to control and visually represent the information sensed by autonomous robots in an intuitive manner. Similarly, in [9] and [34], visual feedback was employed to enhance operator awareness and reduce pilot workload. On the other hand, works such as [20], [15], and [16] focused on simplifying various manipulation activities. Additionally, in [5], visual cues were incorporated to indicate user paths.

¹Eurecat, Centre Tecnològic de Catalunya, Robotics and Automation Unit, Cerdanyola del Vallès, Barcelona, Spain. [riccardo.franceschini, julian.cayero, javier.rodriguez]@eurecat.org

²Automation and Control group, Department of Electrical Engineering, Danish Technical University, Elektrovej, Building 326, DK-2800 Kgs. Lyngby Denmark mafum@elektro.dtu.dk

Paper Video: <https://youtu.be/TvmkFXQlA60>

Concurrently, recent advancements in computer vision, exemplified by Vision Transformer (ViT) [19] and Segment Anything Model (SAM) [17], have significantly elevated image understanding capabilities. SAM’s promptable segmentation prowess, along with its impressive zero-shot capabilities, has unlocked a wide array of potential applications. Researchers have explored SAM’s interaction across domains like medical imaging [31, 30], image editing [33], and Neural Radiance Fields (NeRF) [4], among others. Nevertheless, these models often rely on resource-intensive backbones unsuitable for real-time UAV interactions due to their constrained resources. Thus, researchers have endeavored to enhance real-time suitability through techniques like distillation [32] and architectural variations [35, 36, 13].

Taking inspiration from [3], which employed the concept of interactive object segmentation to achieve accurate detection and manipulation tasks, this paper aims to leverage the latest advancements in promptable segmentation models to enhance the UAV pilot experience in the context of aerial inspection. The objective is to introduce an innovative interaction framework that, to the best of our knowledge, has not been published yet, aiming to simplify the task of visually defining and inspecting surfaces. This approach allows the operator to maintain control throughout the entire operation while harnessing the capabilities of a segmentation model to define detailed inspection areas, eliminating the need for laborious manual area definition. Interactive segmentation is seamlessly integrated with an XR interface, which through a monitor allows real-time visualization and control of the proposed regions. The segmented regions are subsequently linked with a planner that utilizes the extracted spatial segmentation to generate a path to cover them. To close the interaction loop, it is also proposed to integrate the SONY DualSenseTM Wireless Controller [8], although other options such as RC controllers could be considered. This controller facilitates tasks like defining areas, controlling the path, and taking over control when necessary. The proposed framework is validated within the Flightmare [27] photorealistic graphic simulator, utilizing ROS [28], Gazebo [18], and RotorS [10] for physics simulation. Further evaluation is also proposed to showcase its ability to run on a real platform with limited processing capabilities.

II. METHODOLOGY

This section provides an overview of the methodology used for detection, planning, and the distribution of control between the UAV and the operator. It is assumed that the UAV has the ability to perceive its three-dimensional envi-

ronment, estimate odometry, and orient its camera toward any direction, either through its omnidirectional capabilities [12] or through vision systems equipped with gimbal or fisheye cameras capable of reaching any angle [11]. The following sections will begin by covering the detection procedure (Section II-A), followed by the planning process (Section II-B). Subsequently, an architecture for integrating the operator into the planning loop is proposed (Section II-C), and finally, the visualization technique is expounded upon (Section II-D).

A. Detection

The current section focuses on the detection pipeline, which encompasses the transition from 2D extraction to 3D data manipulation. For the sake of simplicity, it is assumed that the information collected from the camera sensor also has an equivalent in spatial data, thus enabling the retrieval of its position in space. The proposed framework, is model agnostic as long the model can be prompted by positive and negative point definition. Depending on computation availability the segmentation model can be interchanged based on the use case, in this paper we deployed HQ-SAM [13] in the simulation environment, which uses TinyViT [32] variant as its backbone. This selection is due to its capacity to maintain highly detailed segmentation masks while significantly reducing computational costs compared to the original backbone. While for the deployment on embedded hardware with limited capability FastSAM [36] is chosen. Thus, to query the model, consider a point \mathbf{p} in 2D space, expressed as $\mathbf{p} = (x, y)$, where x and y are the cartesian coordinates of the point over the image plane. Furthermore, consider a label l belonging to the set $\{0, 1\}$, where 0 signifies negative points and 1 corresponds to positive ones. This label l is assigned to a point \mathbf{p} , indicating 1 if there is an area that should be included in the segmentation mask and 0 for areas that should be excluded. Consequently, the model is fed with a collection of points along with their corresponding labels, denoted as P . Each element within P comprises a pair consisting of a point \mathbf{p} and its associated label l :

$$P = \{[\mathbf{p}_0, l_0], [\mathbf{p}_1, l_1], \dots, [\mathbf{p}_n, l_n]\} \quad (1)$$

Following the indication from P , the network retrieves a mask, from which contours are extracted using OpenCV [2]. These contours are then subjected to filtration, retaining only the primary contours. Given the satisfactory outcome of the 2D segmentation process, the identified polygon serves as the foundation for extracting the desired point cloud, denoted as $\mathcal{P} = \{\mathbf{x}_i \mid i = 1..N\}$, where each point $\mathbf{x}_i \in \mathbb{R}^3$ is retrieved from depth information by extracting the points that are within the extracted polygon and converted to 3D points using the camera information matrix.

Initially, the points undergo voxelization and are subjected to geometrical outlier removal, following algorithms outlined in [37]. This step aims to eliminate any potential errors stemming from the segmentation process that might result in extraneous points. Then, the normal vectors $\mathbf{n}_i = \{x_i, y_i, z_i\}$ of \mathcal{P} are derived using covariance analysis. It is important to note that these normals might exhibit opposing directions.

Consequently, an adjustment is performed to ensure that all normals are oriented to face the camera frame. Subsequently, the designated inspection area is forwarded to the planning pipeline for further processing.

B. Planning

Assuming the presence of a segmented surface represented by \mathcal{P} . The objective of the planner is to generate an arrangement of points $\mathbf{k}_i \in \mathbb{R}^3$, positioned at a specified distance d_s from the surface into the direction of the surface normal \mathbf{n}_i and with camera orientation $\mathbf{o}_i \in \mathbb{R}^3$ perpendicular to the surfaces such as $\mathbf{o}_i = -\mathbf{n}_i \forall i$. Consequently, a path can be formulated as follows:

$$\mathcal{K} = \{(\mathbf{k}_0, \mathbf{o}_0), (\mathbf{k}_1, \mathbf{o}_1), (\mathbf{k}_2, \mathbf{o}_2), \dots, (\mathbf{k}_n, \mathbf{o}_n)\} \quad (2)$$

Thus, as starting point to ensure coverage of the surface the voxelized points in \mathcal{P} are used. Then, each point is translated by a distance of d_s along the normal \mathbf{n}_i :

$$\mathbf{k}_i = \mathbf{p}_i + d_s \mathbf{n}_i \quad (3)$$

Through this process, an unordered sequence of waypoints \mathcal{K} is generated. To address this, the problem is reformulated into a classical Traveling Salesman Problem (TSP), with the goal of minimizing the total path length required to traverse the entire sequence of waypoints. Although the TSP is known to be NP-hard, the number of waypoints in this context is limited and there is no constant need to dynamically update the path. Hence, the 2-opt algorithm [6] is employed to retrieve the ordered sequence \mathcal{K}^* . This local search optimization technique starts with an initial tour and repeatedly improves it by swapping two edges to reduce tour length until no more improvements can be made. While not guaranteed to find the optimal solution, it is a computationally efficient method for obtaining high-quality TSP solutions. A visual example of \mathcal{K}^* at distance d_s with respect to a detected area \mathcal{P} is reported in Fig.2

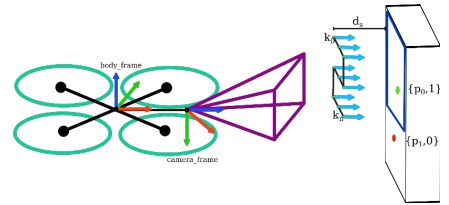


Fig. 2: Visual representation of UAV reference frames, \mathcal{K}^*

C. Remote Control

Efficient Human-Robot Interaction (HRI) hinges on a crucial factor: the device employed for communicating with the robot. In this context, the approach presented in this paper integrates a PS5 controller. This choice is motivated by the controller's widespread commercial availability and its customizable flexibility. In Fig. 3, the configuration for interacting with the UAV is depicted. The UAV orchestrates

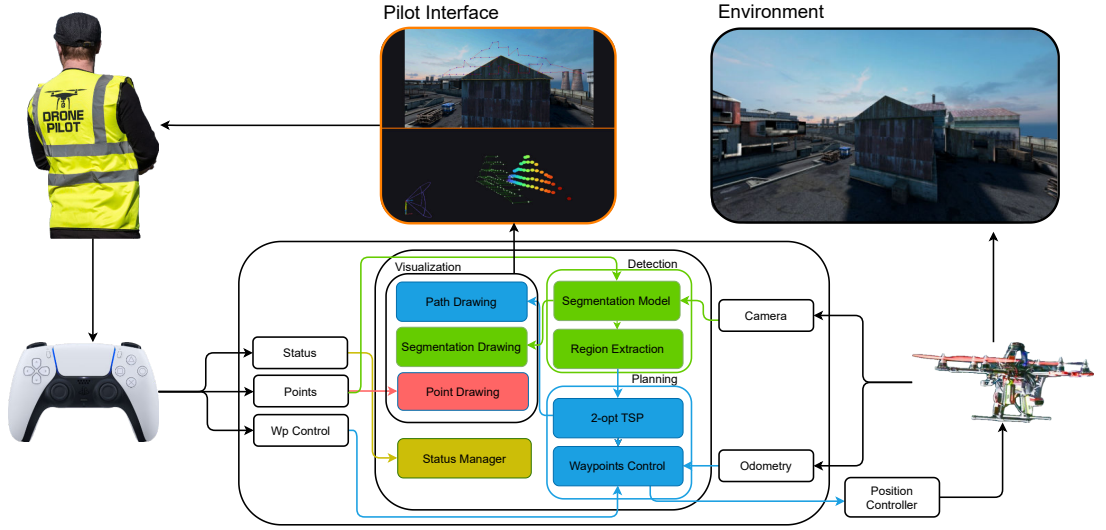


Fig. 1: The provided diagram illustrates the interaction dynamics between the operator and the UAV. More precisely, the suggested module receives input points denoted as P , status indications marked as S , or manual waypoint commands issued by the operator. Concurrently, it acquires odometry data and camera information from the UAV. Based on the operator's specified status, the proposed layer employs a segmentation model to segment the camera data, utilizing the operator's input points (P). Subsequently, it undertakes the path planning across the identified area, effectively addressing the Traveling Salesman Problem (TSP) with the local search 2-opt algorithm. The module then manages waypoints, either through manual directives or autonomous procedures. Running in parallel, the visualization module is responsible for upholding a comprehensive situational awareness for the operator. This is achieved by displaying a range of information on the user interface. This information dynamically transitions from interactive points and segmentation masks during the segmentation phase to proposed 2D and 3D plans during the planning and waypoint following stages.

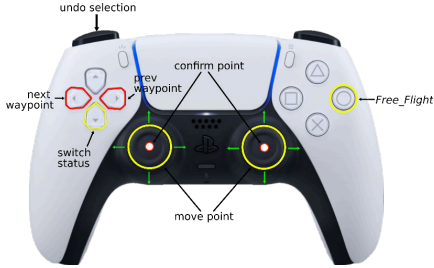


Fig. 3: Controller Interaction

various phases of the inspection, including detection, planning, and movement through a series of waypoints. These phases are encoded in different states, which are as follows:

$$S = \{ \text{Free_flight}, \\ \text{Aim_To_Surface}, \\ \text{Plan_To_Surface}, \\ \text{Auto_Move_To_Surface}, \\ \text{Manual_Move_To_Surface} \}$$

To facilitate effective interaction with the system, the operator must have the capability to seamlessly transition between those states. The *Free_flight* mode is initiated by pressing the circle button. This mode allows the operator to assume control of the UAV at any time, granting the complete authority over the UAV as they would have in a normal position control mechanism. Subsequently, by pressing the arrow down button, the *Aim_To_Surface* mode is activated. During this phase, the operator defines positive and negative points denoted as P on the image. To achieve this, the operator utilizes the left and right thumbsticks to control positive and negative pointers, which are consistently displayed over the camera feedback. Confirming each point is as simple as pressing the opposite thumbstick, while undoing a point requires pressing the left trigger. This process provides the model with the necessary labels for performing segmentation. Once the operator is satisfied with the desired segmentation, pressing the arrow down button again activates the *Plan_To_Surface* state. In this state, the path (K^*) is retrieved and displayed on the operator interface. If the operator approves the retrieved path, pressing the arrow down button advances the state to *Auto_Move_To_Surface*, initiating the UAV's automatic movement along the retrieved waypoints by sending it to the next waypoint if the UAV's position is within a radius

r from the current waypoint. Whenever the operator wishes to assume control over the movement along the waypoints, pressing the arrow down button once more enables the *Manual_Move_To_Surface* mode. In this mode, using the left and right arrow buttons, the operator can navigate the waypoints within \mathcal{K}^* , moving both forwards and backwards. Continuing with the downward arrow progression, the UAV will alternate between the *Auto_Move_To_Surface* and *Manual_Move_To_Surface* modes until the UAV either completes traversing \mathcal{K}^* or the operator presses the circle button. A schematic representation of the state machine is reported in Fig. 4.

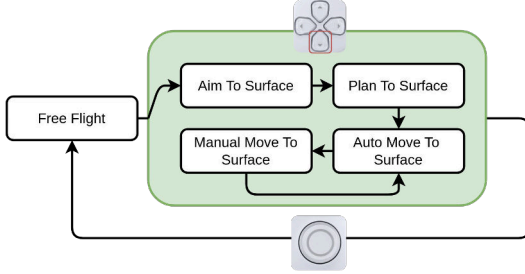


Fig. 4: This scheme represents the transition between different states (S), with the downward arrow button responsible for switching between states and a circle used to regain control

D. Visual Feedback

Ensuring the pilot's continuous involvement and awareness during flight operations is of utmost importance. This requires real-time communication of information and ongoing processes within the UAV to the pilot via their visualization device, which can take the form of a tablet or monitor. In the proposed solution, there are three key aspects that must be visually presented to keep the operator informed and engaged. Firstly, in the *Aim.To.Surface* state, when the segmentation area is being defined, the user observes two distinct points: a green point and a red point. These points symbolize the positive and negative points intended for transmission as prompts to the model. As detailed in Section II-C, these points are manipulated and continually updated using the controller. Upon confirmation by pressing the thumbstick, they become permanently marked on the image. Simultaneously, the segmentation area is consistently outlined and revised in accordance with the user-defined points P , as reflected in the camera feedback Fig 5. This multifaceted visualization approach ensures that the pilot remains informed about the ongoing processes within the UAV and allows for active participation in decision-making. Within this illustration, the two green dots situated at the upper section of the building delineate the points P . Beneath them, another pair of dots, one green and one red, signifies the thumbstick-controlled pointing dots. Additionally, the contour of the selected surface is highlighted to draw the

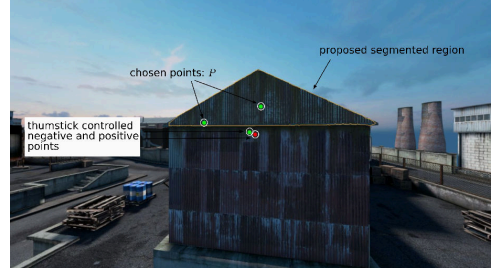


Fig. 5: P definition and visualization on camera stream

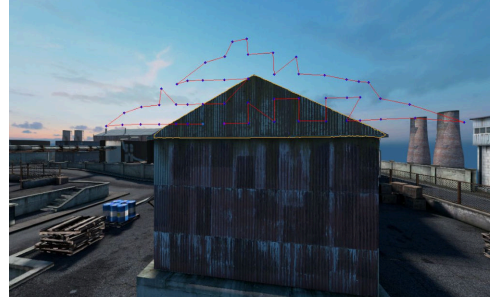


Fig. 6: Path Visualization on camera feedback

operator's attention to the proposed region. If the operator find the segmentation satisfactory and opt to switch to the *Plan.To.Surface* state, the path is visualized as depicted in Fig. 6. This visualization is also reproduced in the 3D representation shown in Fig. 7, where comprehensive orientation information of the drone is also provided. These representations serve to provide the operator with a clear understanding of the spatial aspects and assist in decision-making and operational coordination. Subsequently, upon transitioning to either the *Auto.Move.To.Surface* or *Manual.Move.To.Surface* state, the camera stream presents a progress indicator situated at the top right corner, displaying the percentage of completion along \mathcal{K}^* (Fig. 8). This feature empowers the operator to retain control over the surveyed surface while being informed of the ongoing execution progress. Concurrently, the 3D visualization (Fig 7) undergoes updates to reflect the orientation and position of the drone as it traverses \mathcal{K}^* . This real-time representation provides a dynamic overview

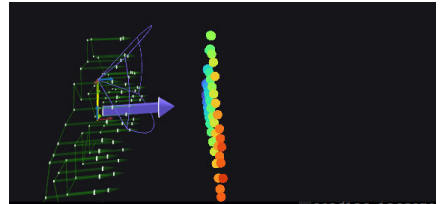


Fig. 7: Visualization of the segmented surface and \mathcal{K}^*



Fig. 8: Visualization of the path completion K^*

of the drone's movement and orientation, contributing to the operator's comprehensive understanding and effective maneuvering throughout the operation.

III. EVALUATION IN SIMULATION

This section is dedicated to evaluating the performance of the proposed approach when segmenting and inspecting various surfaces. The assessment was conducted within the Flightmare [27] photorealistic graphics simulator, making use of ROS [28], Gazebo [18], and RotorS [10] for physics simulation. The objective is to ascertain whether the integration of a segmentation model can enhance the operator's experience by minimizing the number of inputs needed to define intricate areas. To achieve this, the evaluation has been conducted within a simulated industrial environment accessible through Flightmare. This environment is particularly suitable for such tests due to its photorealistic nature and the availability of infrastructures akin to those encountered by an operator during daily tasks. The experiments were conducted using a GTX1080ti [24], enabling HQ-SAM to process images at a resolution of 1280×720 at approximately 7Hz. This, thanks to the decoupled segmentation and visualization processes, allowed for a seamless interaction with the operator. To assess performance, different structures, such as tanks, containers, roofs, and chimneys, were evaluated as depicted in Fig. 9. These images demonstrate the capability to accurately segment intricate structures with minimal user intervention. In the provided example, a maximum of four points (P) – encompassing negative and positive instances – were required. Furthermore, the resulting segmented clouds and paths are illustrated in Figs. 9i, 9j, 9k and 9l. The extracted set of points (P) are also evaluated, showcasing the filtering procedure's efficacy in isolating the desired points. As a result, the generated paths (K^*) consistently align with the surfaces. Thus if the operator is satisfied with the retrieved path can proceed and change the status to *Auto_Move_To_Surface* and start the inspection, on the other hand if the segmentation is unsuccessful the operator is always in control and able to change and move back.

IV. REAL-WORLD EVALUATION

To further validate the performance of the interaction process, a real platform test was conducted. This test utilized an FPV-like drone equipped with a Voxl2 [23] board, which

integrates a Qualcomm QRB5165 [26] SoC capable of performing onboard inference for deep learning models at 15 TOPS. This setup enabled us to deploy a quantized version of FastSAM [36], facilitating onboard inference at a rate of 5Hz. Thanks to the decoupled visualization and inference pipeline, we were able to maintain a video feedback at 30Hz on the user interface. The flight controller PX4 [22] is integrated within the board, while a RealSense d435 [14] serves as the primary depth sensor. The interaction pipeline was tested in two distinct scenarios: a pipe-like structure and a flat wall surface with a patch to be segmented (Fig. 10).

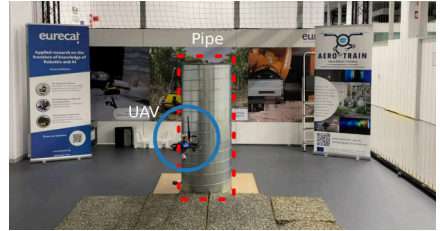


Fig. 10: Experiment Setting

As shown in Fig. 11, the operator can delineate regions by specifying a single positive point (Fig. 11a and 11b).

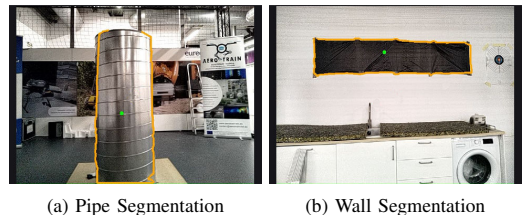


Fig. 11: Point Segmentation and Plan of a pipe

The surface extraction results are presented in Fig. 12 for the pipe scenario and Fig. 13 for the wall patch scenario. These plots illustrate the executed trajectory together with the segmented point cloud, demonstrating the system's capability to accurately extract the desired object and follow a consistent orientation aligned with the inspected surface.

V. CONCLUSION

In conclusion, the framework presented in this paper introduces an innovative approach to enhance HRI while teleoperating UAV for surface analysis. In this approach, the operator takes an active role in the decision-making process by defining points of interest, which are used by the segmentation model to perform RGB segmentation. The segmented area is then processed through a point cloud extraction pipeline to retrieve the spatial data and eliminate undesired elements from the RGB segmentation. Subsequently, the refined segmentation data is employed in a local search algorithm to efficiently solve the Traveling Salesman

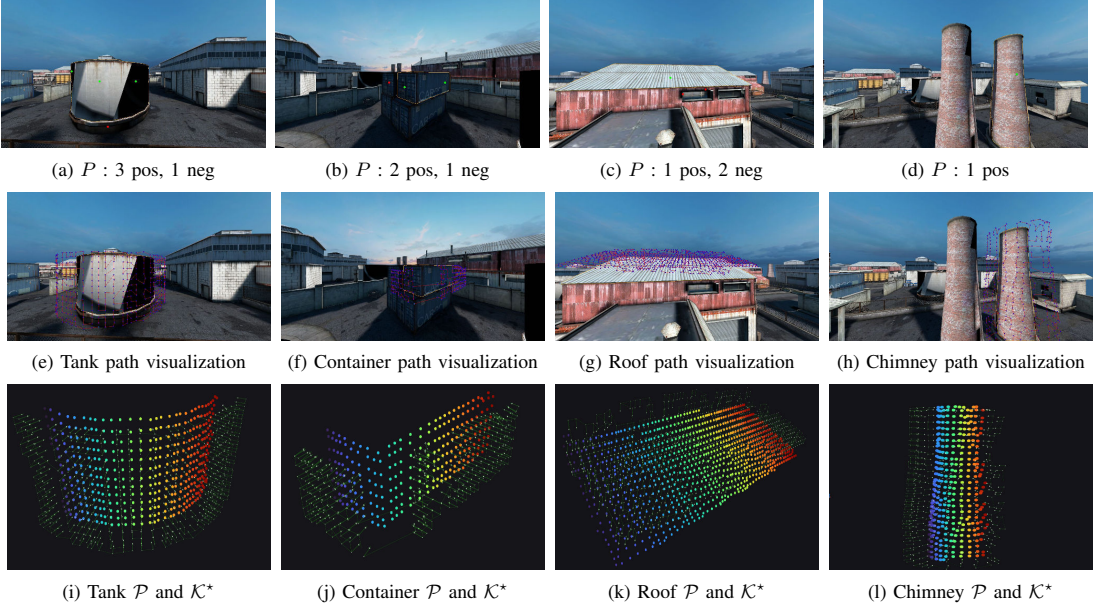


Fig. 9: Anecdotal performance comparison involving various inspection surfaces, the first row displays the operator-selected points and the extracted mask, the second row highlights the superimposed path, while the last row presents the extracted spatial data alongside the path.

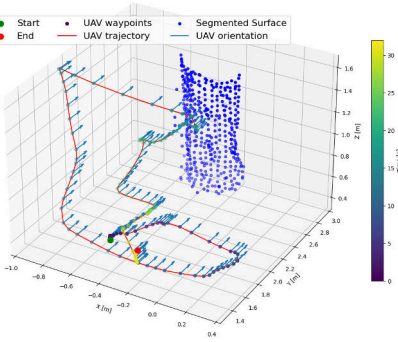


Fig. 12: Trajectory while inspecting a pipe

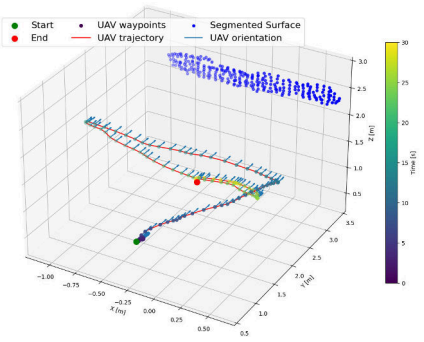


Fig. 13: Trajectory while inspecting a wall

Problem (TSP), generating a path across the segmented spatial data. Throughout this process, the operator remains informed through visual cues, which include points of interest, segmented surfaces, generated paths, and execution progress indicators. Concurrently, control remains with the operator, allowing to define points of interest, manage the drone's status and movement along the path. The proposed methodology has undergone validation within both simulated environments and real-world laboratory setups, demonstrating its capacity to adapt to different environments and run on embedded hardware at a satisfactory inference rate, allowing the op-

erator to define complex inspection regions with minimal interaction. It is important to highlight that this integration takes place without imposing any additional complexities, such as extra devices or interactions that could potentially compromise operational safety. Moving forward, the next steps will involve field evaluation, analyzing potential issues due to sensor noise and communication delays.

ACKNOWLEDGMENT

This work has been supported by the European Unions Horizon 2020 Research and Innovation Programme AERO-TRAIN under Grant Agreement No. 953454 and by the

Catalan Government through the funding grant ACCIÓ-Eurecat (Project TRAÇA – EUTFS).

REFERENCES

- [1] Jacopo Aleotti et al. “Detection of Nuclear Sources by UAV Teleoperation Using a Visuo-Haptic Augmented Reality Interface”. In: *Sensors* 17.10 (2017). ISSN: 1424-8220. DOI: 10 . 3390 / s17102234. URL: <https://www.mdpi.com/1424-8220/17/10/2234>.
- [2] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).
- [3] Daniel J. Butler, Sarah Elliot, and Maya Cakmak. “Interactive scene segmentation for efficient human-in-the-loop robot manipulation”. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 2572–2579. DOI: 10 . 1109/IROS.2017.8206079.
- [4] Xiaokang Chen et al. “Interactive Segment Anything NeRF with Feature Imitation”. In: *arXiv preprint arXiv:2305.16233* (2023).
- [5] Andre Cleaver et al. “Dynamic Path Visualization for Human-Robot Collaboration”. In: *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*. HRI ’21 Companion. Boulder, CO, USA: Association for Computing Machinery, 2021, pp. 339–343. ISBN: 9781450382908. DOI: 10.1145/3434074.3447188. URL: <https://doi.org/10.1145/3434074.3447188>.
- [6] G. A. Croes. “A Method for Solving Traveling-Salesman Problems”. In: *Operations Research* 6.6 (1958), pp. 791–812. ISSN: 0030364X, 15265463. URL: <http://www.jstor.org/stable/167074> (visited on 08/16/2023).
- [7] Jeffrey Delmerico et al. “Spatial Computing and Intuitive Interaction: Bringing Mixed Reality and Robotics Together”. In: *IEEE Robotics Automation Magazine* 29.1 (2022), pp. 45–57. DOI: 10.1109/MRA.2021.3138384.
- [8] Sony Interactive Entertainment. *DualSense Wireless Controller*. <https://www.playstation.com/en-us/accessories/dualsense-wireless-controller/>. 2021.
- [9] Riccardo Franceschini, Matteo Fumagalli, and Julian Cayero Becerra. “Enhancing Human-Drone Interaction with Human-Meaningful Visual Feedback and Shared-Control Strategies”. In: *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2023, pp. 1162–1167. DOI: 10 . 1109 / ICUAS57906.2023.10156190.
- [10] Fadri Furrer et al. “Robot Operating System (ROS): The Complete Reference (Volume 1)”. In: ed. by Anis Koubaa. Cham: Springer International Publishing, 2016. Chap. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. ISBN: 978-3-319-26054-9. DOI: 10 . 1007 / 978-3-319-26054-9_23. URL: http://dx.doi.org/10.1007/978-3-319-26054-9_23.
- [11] Andreas Humpe. “Bridge inspection with an off-the-shelf 360° camera drone”. In: *Drones* 4.4 (2020), p. 67.
- [12] Mina Kamel et al. “The Voliro Omniorientational Hexacopter: An Agile and Maneuverable Tilttable-Rotor Aerial Vehicle”. In: *IEEE Robotics & Automation Magazine* 25.4 (Dec. 2018), pp. 34–44. DOI: 10 . 1109/mra.2018.2866758. URL: <https://doi.org/10.1109%2Fmra.2018.2866758>.
- [13] Lei Ke et al. “Segment Anything in High Quality”. In: *arXiv:2306.01567* (2023).
- [14] Leonid Keselman et al. “Intel RealSense Stereoscopic Depth Cameras”. In: *CoRR* abs/1705.05548 (2017). arXiv: 1705.05548. URL: <http://arxiv.org/abs/1705.05548>.
- [15] Dongbin Kim and Paul Y. Oh. “Aerial Manipulation using a Human-Embodied Drone Interface”. In: *2022 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*. 2022, pp. 1–7. DOI: 10.1109/ARSO54254.2022.9802972.
- [16] Dongbin Kim and Paul Y. Oh. “Toward Avatar-Drone: A Human-Embodied Drone for Aerial Manipulation”. In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2021, pp. 567–574. DOI: 10 . 1109/ICUAS51884.2021.9476704.
- [17] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: 2304.02643 [cs.CV].
- [18] N. Koenig and A. Howard. “Design and use paradigms for Gazebo, an open-source multi-robot simulator”. In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.
- [19] Alexander Kolesnikov et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: 2021.
- [20] Tsung-Chi Lin, Achyuthan Unni Krishnan, and Zhi Li. “Comparison of Haptic and Augmented Reality Visual Cues for Assisting Tele-manipulation”. In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 9309–9316. DOI: 10 . 1109/ICRA46639.2022.9811669.
- [21] Chuhao Liu and Shaojie Shen. “An Augmented Reality Interaction Interface for Autonomous Drone”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 11419–11424. DOI: 10 . 1109 / IROS45743 . 2020 . 9341037.
- [22] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. “PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 6235–6240. DOI: 10.1109/ICRA.2015.7140074.

- [23] ModalAI. *VOXL2: A Powerful Companion Computer for Drones*. <https://www.modalai.com/products/voxl-2?variant=39914779836467>. Accessed 2024-02-08.
- [24] NVIDIA Corporation. *GeForce GTX 1080 Ti*. [GPU]. 2017. URL: <https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1080-ti/specifications/>.
- [25] F.J. Perez-Grau et al. "Semi-autonomous teleoperation of UAVs in search and rescue scenarios". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2017, pp. 1066–1074. DOI: 10.1109/ICUAS.2017.7991349.
- [26] Qualcomm Technologies, Inc. *QRB5165 SoC Product Brief*. https://www.qualcomm.com/content/dam/qcomm-martech/dm-assets/documents/qrb5165-soc-product-brief_87-28730-1-b.pdf. Year of Access.
- [27] Yunlong Song et al. "Flightmare: A Flexible Quadrotor Simulator". In: *Conference on Robot Learning*. 2020.
- [28] Stanford Artificial Intelligence Laboratory et al. *Robotic Operating System*. Version ROS Melodic Morenia. May 23, 2018. URL: <https://www.ros.org>.
- [29] Ryo Suzuki et al. "Augmented Reality and Robotics: A Survey and Taxonomy for AR-Enhanced Human-Robot Interaction and Robotic Interfaces". In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI '22. New Orleans, LA, USA: Association for Computing Machinery, 2022. ISBN: 9781450391573. DOI: 10.1145/3491102.3517719. URL: <https://doi.org/10.1145/3491102.3517719>.
- [30] Junde Wu. "PromptUNet: Toward Interactive Medical Image Segmentation". In: *arXiv preprint arXiv:2305.10300* (2023).
- [31] Junde Wu et al. *Medical SAM Adapter: Adapting Segment Anything Model for Medical Image Segmentation*. 2023. arXiv: 2304.12620 [cs.CV].
- [32] Kan Wu et al. *TinyViT: Fast Pretraining Distillation for Small Vision Transformers*. 2022. arXiv: 2207.10666 [cs.CV].
- [33] Jingfeng Yao et al. *Matte Anything: Interactive Natural Image Matting with Segment Anything Models*. 2023. arXiv: 2306.04121 [cs.CV].
- [34] Mohammad Kassem Zein et al. "Deep Learning and Mixed Reality to Autocomplete Teleoperation". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 4523–4529. DOI: 10.1109/ICRA48506.2021.9560887.
- [35] Chaoning Zhang et al. "Faster Segment Anything: Towards Lightweight SAM for Mobile Applications". In: *arXiv preprint arXiv:2306.14289* (2023).
- [36] Xu Zhao et al. *Fast Segment Anything*. 2023. arXiv: 2306.12156 [cs.CV].
- [37] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

AeroAssistant: A Modern and Flexible UAV Teleoperation Framework

RICCARDO FRANCESCHINI*, Eurecat, Centre Tecnològic de Catalunya, Robotics and Automation Unit, Spain

MATTEO FUMAGALLI, Automation and Control group, Department of Electrical Engineering, Danish Technical University, Elektrovej, Lyngby, Denmark

JULIAN CAYERO BECERRA, Eurecat, Centre Tecnològic de Catalunya, Robotics and Automation Unit, Spain

Operating unmanned aerial vehicles (UAVs) is a non-trivial task, particularly in the context of asset inspections, where pilots encounter unique challenges. These include navigating in cluttered and hazardous environments, maintaining a safe distance from the inspection area, ensuring correct orientation towards the inspected surface, and achieving comprehensive coverage of the entire surface. Achieving these tasks is inherently complex and stressful even for expert pilots. To enhance the navigation experience, this work introduces AeroAssistant, a flexible teleoperation framework designed to facilitate drone operations through a combination of shared control algorithms and augmented reality visual cues. The primary objective of AeroAssistant is to propose a framework capable of providing an intuitive and familiar teleoperation interface, which incorporates a system of plugins capable of simplify complex and tedious tasks. This paper delineates the structural components of the framework, elucidates interaction paradigms, and outlines the features currently integrated and deployed in real UAV systems within the AeroAssistant framework.

CCS Concepts: • **Human-centered computing** → *Interaction techniques*; **Collaborative interaction**.

Additional Key Words and Phrases: HRI, Augmented Reality, Shared Autonomy, Unmanned Aerial Vehicles

ACM Reference Format:

Riccardo Franceschini, Matteo Fumagalli, and Julian Cayero Becerra. 2024. AeroAssistant: A Modern and Flexible UAV Teleoperation Framework. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 18 pages. <https://doi.org/XXXXXXX.XXXXXXX>

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs), thanks to their ability to move freely in space and reach remote areas, have emerged as versatile tools in a variety of scenarios ranging from search and rescue operations to industrial inspections. In particular, for industrial inspection, UAVs are utilized to manage critical infrastructure in our society, including roads [36], bridges [24, 41], power lines [42], and wind turbine facilities [19]. The integrity of the infrastructure and the safety of the operation are of utmost importance, where direct teleoperation by the operator remains the dominant control paradigm. Therefore, collaborative efforts between drones and operators are essential to ensure safety, particularly in tasks demanding precise navigation within challenging and critical environments, as emphasized in [1, 37]. To enhance human-robot interaction, various methods have been proposed in the literature studying how extended reality

*also with Automation and Control group, Department of Electrical Engineering, Danish Technical University, Elektrovej, Lyngby Denmark

Authors' Contact Information: Riccardo Franceschini, riccardo.franceschini@eurecat.org, Eurecat, Centre Tecnològic de Catalunya, Robotics and Automation Unit, Cerdanyola del Vallès, Barcelona, Spain; Matteo Fumagalli, mafum@dtu.dk, Automation and Control group, Department of Electrical Engineering, Danish Technical University, Elektrovej, Lyngby, Lyngby, Denmark; Julian Cayero Becerra, julian.cayero@eurecat.org, Eurecat, Centre Tecnològic de Catalunya, Robotics and Automation Unit, Cerdanyola del Vallès, Barcelona, Spain.

This article has been submitted to ACM Transactions on Human-Robot Interaction for consideration. Its contents may not be further disseminated until a final decision regarding publication has been made and further permission has been granted.

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

Manuscript submitted to ACM

(XR) and shared-control paradigms are used to ease collaboration by reducing friction in controlling even the most complex systems while increasing operator situational awareness [43]. A more in-depth evaluation specifically about UAV teleoperation is carried out in Section 2. However, to the best of our knowledge, a unified system capable of integrating multiple shared interaction control algorithms, defined herein as flight plugins, alongside XR interfaces while preserving a familiar pilot experience, has yet to be proposed. Therefore, the objective of this work is to introduce AeroAssistant, a teleoperation framework designed to enhance the operator’s pilot experience by proposing a series of plugins aimed at simplifying tedious and complex navigation tasks common during operation. The framework is engineered with efficiency and flexibility in mind, employing a multithreaded architecture capable of maintaining a satisfactory user experience even with constrained resources. This architecture abstracts the core functionalities from the message-passing requirements, enabling the use of the same codebase with different communication frameworks such as ROS1 [39], ROS2 [29], or other message-passing libraries like ZMQ [17]. The plugins aim to streamline the teleoperation experience through a combination of enhanced user interfaces and shared control algorithms while maintaining a familiar teleoperation control scheme with a commercial controller. Thus, the paper is structured as follows: first, an overview of state of the art is proposed in Section 2, then the AeroAssistant architecture is described in detail in Section 3, while a description of the current capabilities in terms of plugins showcased in Section 4.

2 RELATED WORK

The landscape of human-robot interaction, especially in teleoperation scenarios, has seen a variety of approaches aimed at improving user experience and system performance. Numerous studies have explored various strategies, such as XR interfaces, shared control mechanisms employing haptic devices, integration of deep learning agents, as well as the utilization of path planning and collision avoidance systems. Specifically focusing on drone interaction, these approaches can be categorized broadly in different high-level categories.

Assisted Standard Pilot Interfaces: Systems like those discussed in [35, 34, 2] involve the operator managing the UAV’s velocity similar to standard teleoperation. Collision avoidance algorithms are implemented between the operator and the UAV to adjust the operator’s commands if they pose a collision risk. Another relevant form of assisted teleoperation is studied in [8]. In this study, the authors explored teleoperation constrained to virtual surfaces, where the operator defines geometric surfaces that restrict the UAV’s movement. This setup allows the pilot to control the UAV’s movement simply by adjusting its position along these virtual surfaces.

Augmented Reality Interfaces for Point and Move: Studies like those detailed in [26, 47, 4, 45] present interfaces designed for controlling nearly autonomous drones. These methods necessitate the use of tablet or head-mounted displays (HMDs), with interaction revolving around visualizing 3D data collected by the drone or the drone’s trajectory overlaid on a physical model in front of the operator. The core interaction involves defining a destination point and visualizing the UAV movement in space through the proposed interfaces representing a miniaturized world.

Haptic Control for Trajectory Manipulation: Other approaches, such as those explored in [3, 30, 28, 46, 31, 51], focus on utilizing haptic devices for modifying trajectories, enhancing control experiences by offering various haptic feedback schemes to users. These methods aim to involve the operator in the UAV control process, enabling them to perform vertical and lateral displacements along a predefined trajectory while providing haptic feedback based on obstacle and platform limitations.

Deep Learning Teleoperation: Noteworthy teleoperation methodologies are proposed in [50, 49], where authors propose learning common trajectories, such as turns, straight lines, or circular shapes, using a deep learning agent that

interprets the operator's control input. The agent is then positioned between the operator and the UAV, taking control when it recognizes the operator's attempt to execute one of these trajectories, thereby executing precise movements.

Gaze and Body Control Navigation: Intriguing interaction methods are also introduced in [48, 11], where the operator's gaze is detected via an HMD and used to discern the operator's intention, adjust the UAV trajectory, or designate a point for UAV movement. Additionally, other studies such as [15] have explored using body gestures to control drone movement.

XR assisted navigation: Other studies have investigated the use of enhanced visualization to improve the navigation experience. For instance, [16, 11] demonstrated that employing augmented reality (AR) in a colocated environment, displaying what the drone perceives, effectively improves teleoperation performance. Meanwhile, studies such as [18, 22] have explored the potential of using enhanced remote teleoperation to highlight points of interest or to create a blend of virtual and real elements for a unique flight experience.

AeroAssistant: Our approach introduces a navigation interaction similar to classical teleoperation schemes discussed in the Deep Learning and Assisted pilot methods, while integrating AR elements to enhance operator situational awareness. Furthermore, our proposed framework aims to establish the foundation for a customizable teleoperation experience by offering a unified architecture capable of facilitating shared and visually enhanced interactions.

3 SYSTEM OVERVIEW

For the AeroAssistant to function, it is assumed that the UAV can perceive its three-dimensional environment through either a lidar or a depth camera, has an RGB camera, and can estimate its position and orientation in space $\mathbf{p}_r \in \mathbb{R}^3$, $\mathbf{q}_r \in \mathbb{H}$.

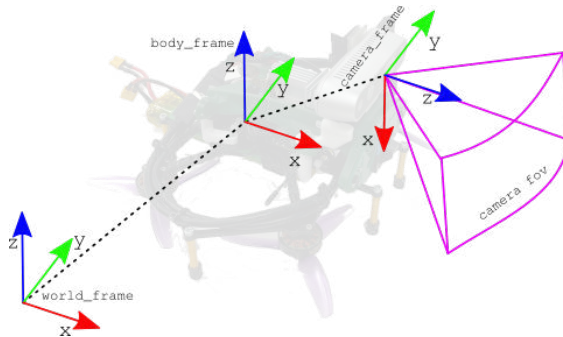


Fig. 1. A schematic representation of the UAV frame configuration

This section aims to explain the framework architecture, focusing on all the components involved in the system. In Fig. 2, an overview schema is presented, consisting mainly of three components: a **RemoteControl** responsible for interpreting user input, the **AeroAssistant** itself, and the **Middleware** responsible for abstracting platform-dependent requirements such as PX4 [32] or DJI [9] from the actual framework. The following sections are then covering the different parts of the system, starting with the orchestration and synchronization of the robot's status among the different elements (3.1). This is followed by a more in-depth explanation of the Middleware (3.2), the RemoteControl (3.3), and finally, an analysis of the core of the AeroAssistant (3.4).

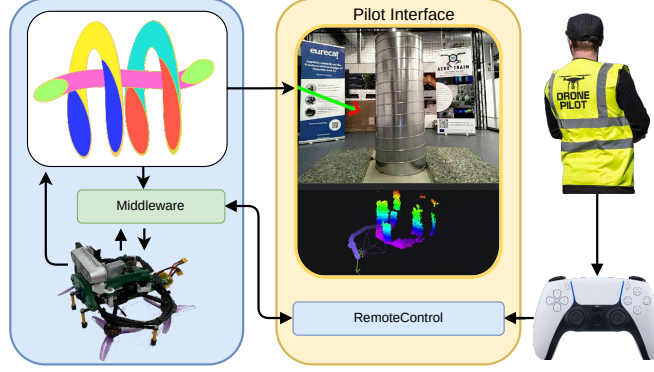


Fig. 2. General framework architecture

3.1 RobotStatus

To ensure coordination within the system and to reflect those changes in both the UAV behavior and the remote pilot interface, it is crucial that all components are aware of the current robot status. Therefore, a state machine capable of encoding the various statuses of the drone is proposed. The robot's status s_r comprises a series of base states \mathcal{S} , which are independent of any plugin implementation, each of which will have its own specific conditions.

$$\mathcal{S} = \{Idle, TakeOff, Land, Free_flight\} \quad (1)$$

These statuses enable pilots to maintain basic control over the UAV, similar to a normal teleoperation mechanism, including autonomous landing and takeoff. To manage synchronization among the various actors of the framework, a request-reply architecture is employed, with the middleware serving as an intermediary between the operator control and the AeroAssistant. This setup ensures message passing and confirmation delivery, as depicted in Figure 3, guaranteeing that all actors are correctly synchronized across different statuses and allowing bidirectional changes in the framework status from both AeroAssistant and RemoteControl. This architecture provides operators with the flexibility to consistently modify states, thereby maintaining full control of the system. At the same time, it empowers the AeroAssistant to perform autonomous behaviors that may involve a sequence of internally coordinated states. These states are promptly communicated to the operator and visualized in the interface, ensuring transparency and awareness.

3.2 Middleware

The interaction middleware operates within the UAV and is responsible for maintaining communication among all involved actors, including the remote pilot input, AeroAssistant, and the UAV itself. Its primary function is to create an abstraction layer, similar to the one proposed in [40], which exposes common topics tied to the specifics of the platform, such as specific routines for arming or sending commands to the platform. This layer is also fundamental for synchronizing communication among all the actors, as explained in Section 3.1. By doing so, it enables both the

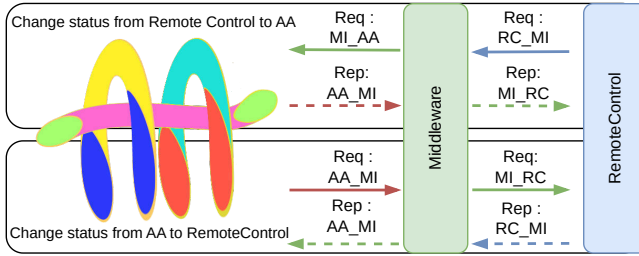


Fig. 3. Status change communication architecture

RemoteControl and the AeroAssistant to publish and subscribe to high-level topics for sending waypoints, arming, or triggering specific routines, while also handling lower-level adjustments specific to the platform under the hood.

3.3 RemoteControl

The RemoteControl aspect of the framework is crucial for ensuring a natural and familiar navigation experience while providing immediate access to the framework's capabilities. For its commercial availability and ease of customization, we utilize the SONY DualSense™ Wireless Controller [10] as a reference, although integration of other RC controllers is feasible without any complications. In order to work the remote controller has to be connected to a companion computer in which run the software for handling all the message passing with the middleware, usually the computer and the receiving device of the operator are the same but that is not an hard constraint. Since the AeroAssistant aim to be a flexible framework with the goal of proposing an enhanced but familiar user experience, the controller needs to change its behaviour depending on the state s_r . Thus by default the operator have the following configuration as in Fig. 4. In this case, the operator can take control and arm the UAV with the left arrow button, which is used to set the

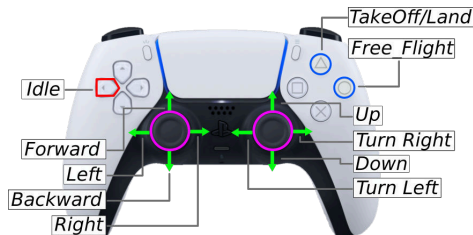


Fig. 4. Standard interaction paradigm

UAV status to $s_r = Idle$. By pressing the triangle button, the status changes to $s_r = TakeOff$, initiating an autonomous takeoff operation at a fixed height h m, defined as a parameter. Once the triangle button is pressed, it will then trigger the status to $s_r = Land$ and initiate autonomous landing by descending. To initiate standard teleoperation in position control, it is necessary to press the circle button, which sets the status to $s_r = Free_Flight$.

3.4 AeroAssistant

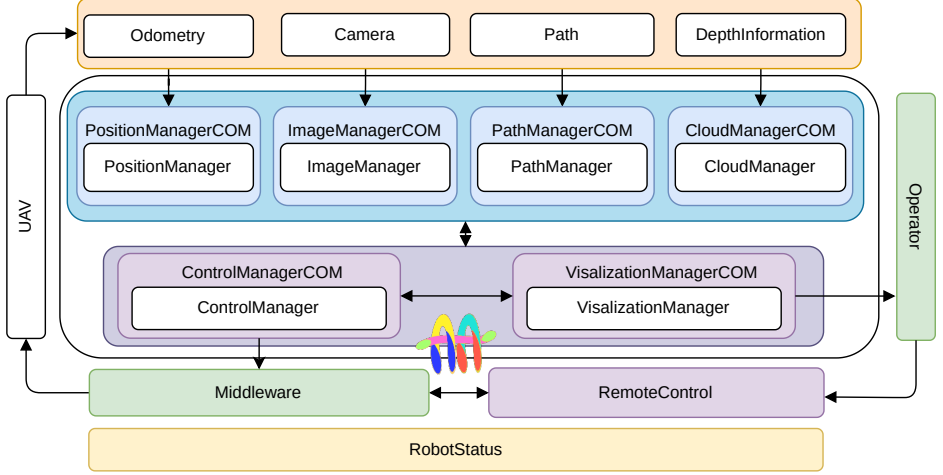


Fig. 5. The provided diagram illustrates the structure of the AeroAssistant, highlighting the interaction and structure between the Middleware, AeroAssistant, and RemoteControl. The AeroAssistant is responsible for receiving, filtering, and exposing information from the UAV, passing it through passive managers to active ones. The diagram also highlights the interaction between the control manager, which controls the platform through the Middleware, and the visualization manager, which maintains an enhanced communication channel with the operator. Throughout all operations, everything is synchronized with the robot status, orchestrating the entire system.

The analysis of the AeroAssistant framework now dive into its core structure, emphasizing its flexibility and adaptability. Given the capability of the UAV to estimate its spatial position and perceive its surroundings, the framework integrates a series of managers tasked with receiving, filtering, and analyzing incoming data. Each manager is encapsulated within a communication wrapper, ensuring interoperability with other system components such as ROS1/2 or ZMQ, tailored to specific system requirements. A detailed overview of the AeroAssistant is depicted in Fig. 5, illustrating its composition with separate managers dedicated to distinct tasks. This architectural approach ensures the framework's autonomy, interoperability, and ease of maintenance. Moreover, these managers provide convenient API calls for interaction with other framework components. Internally, they employ a semaphore multithreading resource-sharing mechanism to securely and efficiently handle processed data. Furthermore, a distinction is made between active and passive managers. Passive managers are primarily responsible for retrieving, processing, and sharing data with other components. Examples include the PositionManager, ImageManager, PathManager, and CloudManager, which receive sensory data from the UAV and additional information such as paths and odometry from external algorithms when internal ones are not employed. On the other hand, the active managers include the ControlManager and the VisualizationManager, which respectively interface with the UAV and the operator interface. The ControlManager interprets user commands by combining information from other managers and transmits commands to the middleware, which subsequently relays them to the UAV. Meanwhile, the Visualization Manager retrieves information from both

Manuscript submitted to ACM

user input and UAV data, combining them into meaningful visualizations for the operator over the camera feedback or spatial representation.

4 PLUGINS

This section aims to expose the plugins currently being deployed within the AeroAssistant, showcasing the interaction paradigms used both from a visualization point of view and RemoteControl perspective. It also describes how S is expanded to include multiple functionalities and how those additional states are used to adjust visualization and RemoteControl behaviors. Therefore, the following section addresses some of the plugins implemented, some of which have already been explained and evaluated in separate publications. For this reason, the evaluation now focuses more on the integration with the framework rather than the functionality and performance itself. The framework has undergone testing on various hardware platforms, including LattePanda Delta3 [25], Intel Nuc [5], and Voxl2 [33]. Despite differences in performance across these platforms, the AeroAssistant consistently managed all computations, maintaining a satisfactory framerate of 30Hz on the operator interface. In the following section, we present data collected from a UAV running the framework on a VOXL2 platform. This platform integrates PX4 [32] onboard and uses a fisheye camera with the Qualcomm Vision SDK [38] to estimate its position $\mathbf{p}_r \in \mathbb{R}^3$ and orientation $\mathbf{q}_r \in \mathbb{H}$. Additionally, a RealSense D435 [21] is used as a depth and visual sensor. The deployed plugins include Lock to a Point Visually (Section 4.1), Move to a Point (Section 4.2), Align and Follow Any Surface (Section 4.3) [12], Rollercoaster (Section 4.4), and Point Segment and Plan (Section 4.5).

4.1 Lock to a point

A common action required when piloting a UAV is the ability to select a point on the camera feedback, lock onto that point, and maintain orientation and position towards it, counteracting possible degradation in position estimation, akin to visual servoing. This feature is integrated within the framework by expanding robot's states S as follows:

$$S = \{S \cup \{\text{Choose_Target}, \text{Following_Target}\}\} \quad (2)$$

An overall control interaction perspective is given in Fig. 6 where the operator (pressing the arrow up) changes the status to $s_r = \text{Choose_Target}$.



Fig. 6. Controller interaction for tracking a point

Next, the operator defines the area to track by moving an overlaid window of dimensions $n \times n$ pixels with the right thumbstick over the camera stream, as highlighted in Fig. 6. Once the area of interest is defined, the operator presses the

arrow up again over the controller to initialize a Kernel Correlation Filter [7] over the previously defined patch. This filter allows tracking of any image patch over the camera stream, as long as enough features can be tracked between consecutive frames. During the operation, the operator receives visual cues overlaid on the camera stream, as depicted in Figure 11, with varying visualizations contingent upon the success or failure of the tracking process. The center of

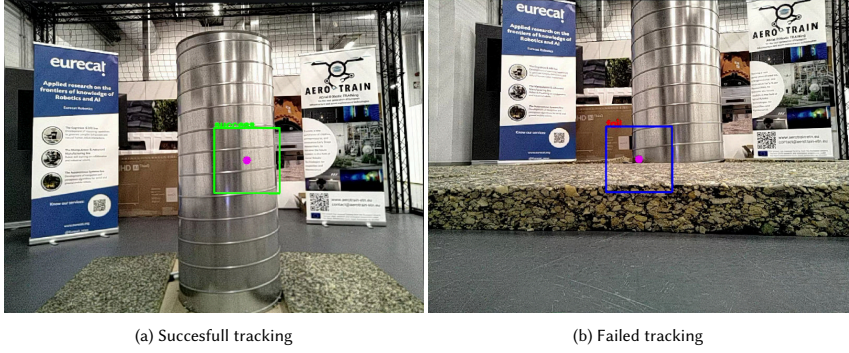


Fig. 7. Different tracking visualizations

the tracked patch is then utilized to extract the position $\mathbf{p}_i \in \mathbb{R}^3$ and normal $\mathbf{n}_i = \{n_{ix}, n_{iy}, n_{iz}\}$ of the point of interest, relying on the depth information retrieved from the previously introduced CloudManager. Subsequently, the point and normal estimations are averaged over k data samples to mitigate potential drift in position and orientation, yielding the averaged values $\bar{\mathbf{p}}_i$ and $\bar{\mathbf{n}}_i$. These averaged values are then utilized to determine the desired position and orientation of the UAV, computed as a position at a distance γ m from the point of interest \mathbf{p}_i

$$\mathbf{p}_d = \bar{\mathbf{p}}_i + \gamma \bar{\mathbf{n}}_i \quad (3)$$

$$\mathbf{o}_d = -\bar{\mathbf{n}}_i \quad (4)$$

and then sent to the UAV position controller to maintain the position \mathbf{p}_r of the UAV in front of \mathbf{p}_i . During the all process the operator is also aware of the surrounding through a 3D panel showcasing the extracted \mathbf{p}_d as shown in Fig. 8.

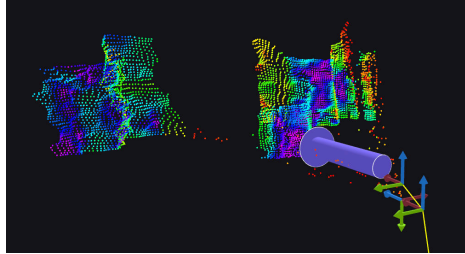


Fig. 8. 3D panel visualization showcasing the extracted tracked point

Moreover the interaction can be improved with the integration of an autonomous detection mechanism, which are responsible of retrieving autonomously the p_i leaving the operator in charge of only initializing the detection and control the correctness of the all approach. A schematic representation of the transition between states is given in Fig. 9

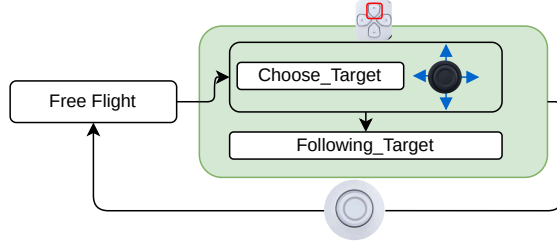


Fig. 9. Transition between different states S for tracking a point

4.2 Move to a point

Another relevant plugin developed is the ability to select a point in the image plane and move towards it, ending up at a user-defined distance γ meters from the obstacle and oriented towards the point of interest. In this case, the robot's state is updated as follows:

$$S = \{S \cup \{Aiming_To_Target, Moving_To_Target\}\} \quad (5)$$

The selection and the change of status to $s_r = Aiming_To_Target$ are triggered by pressing the square button on the controller. In this mode, the operator defines a point $p_m \in \mathbb{R}^2$ on the image plane by using the left thumbstick.



Fig. 10. Controller interaction for moving to a point

As shown in Fig. 10, the operator is presented with a visualization that encodes information about the normal orientation of the surface within the proximity of p_m using a rainbow color-scheme, with p_m itself represented as a yellow dot, as in Fig. 11a. This provides immediate feedback on the geometric structure of the selected area and the feasibility of the selected point.

Then, the point p_i and orientation o_i are extracted as in (3) and (4), respectively, while a planner, in this case, is deployed to guarantee a safe path. The retrieved path is visualized in the 3D panel as shown in Fig. 11b.

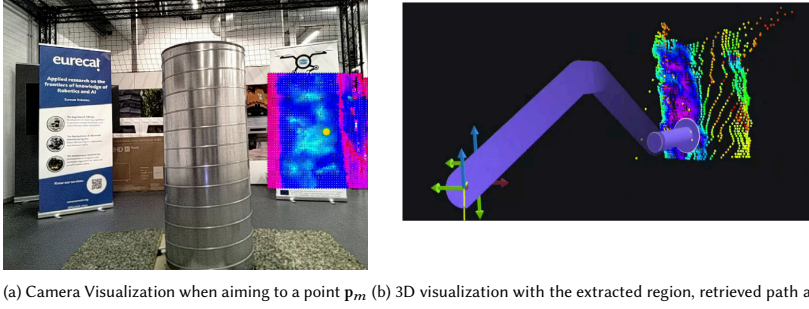
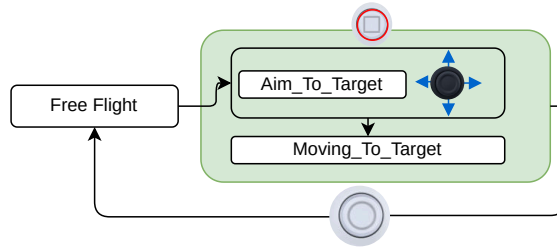


Fig. 11. Different move to a point visualizations

The method introduced in [27] has been adopted as a path planner, primarily due to its efficiency in finding collision-free paths. This efficiency is attributed to its ability to formulate the path generation problem as a quadratic program (QP) featuring the Safe Flight Corridor (SFC) concept. The SFC comprises a collection of convex, overlapping polyhedra that model the available free space and establish a continuous path from the robot's current position to the goal destination. By integrating the SFC, a set of linear inequality constraints is introduced into the QP, thereby facilitating real-time motion planning.

Regarding mapping, the methodology builds upon the research presented in [44]. This work introduces the concept of a sliding window approach to mapping, characterized by the continuous updating of a local map using spatially sensed data. This technique guarantees the availability of a reliable map with sufficient coverage for navigation requirements while simultaneously addressing challenges related to drift and the computational resources demanded by global mapping methods.

Upon pressing the square button again, the state transitions to $s_r = \text{Moving_To_Target}$, initiating the path following procedure. Consequently, the PathManager provides the path, defined as a set of N points $\mathcal{P} = \{p_r, p_0, p_1, \dots, p_m\} \in \mathbb{R}^3$, to the ControlManager. The ControlManager then guides the UAV by transmitting the next waypoint p_{i+1} only when the UAV is within r meters from p_i . A schematic representation of the transition between states is given in Fig. 12

Fig. 12. Move to a point state S transitions.

4.3 Align and Follow Surface

Another relevant feature deployed within AeroAssistant is the ability to align against any surface and maintain distance and attitude facing the surface in front of the UAV, while also being able to perform vertical and lateral displacement. Detailed descriptions of this plugin are provided in previous work [12]. In this work, the focus will be on the integration with the system and modification to fully adapt it with respect to the original work. Thus, to include this behavior within the system, the set of possible statuses is updated as follows:

$$S = \{S \cup \{Align, Lock_To_Surface\}\} \quad (6)$$

The interaction with the remote controller is depicted in Fig. 13. Unlike the previous implementation where the operator triggered functions with a button on a table-like interface, in this case, the operator triggers the following functions with a controller. We argue that this implementation contributes to a more natural experience and interaction with the system compared to the previously published interaction as the operator does not have to switch to another control paradigm. To align with the closest point \mathbf{p}_c the UAV position \mathbf{p}_r , the operator simply presses the cross symbol on the controller, setting $s_r = Align$. This computes the yaw offset η to align with the closest point \mathbf{p}_c and sends the updated orientation to the position controller. During operation, the camera feedback highlights the closest point \mathbf{p}_c as a circle on the camera stream, with colors changing to reflect the angle offset η (green if $|\eta| < 5$, yellow between $5 \leq |\eta| \leq 15$, and red otherwise).



Fig. 13. Control interaction for align and translate

To change the status to $s_r = Lock_To_Surface$, the operator initiates and maintains the status by holding the right trigger on the controller. This automatically stores the current distance $d_{p_c}^t$ to the closest point \mathbf{p}_c at time t , and aligns the UAV to the closest point as described earlier. Unlike the previous work, where translation was binary (either lateral or vertical), the operator now has more fine-grained control. Specifically, both lateral (l_c) and vertical (v_c) displacement are controlled with the thumbstick within the range of $[-1, 1]$ allowing for precision and velocity control over UAV movement. The new desired position is computed as follows. The current position $\mathbf{p}_r^t = [p_x^t, p_y^t, p_z^t]$ at time t is updated as:

$$p_y^{t+1} = p_y^t + \delta l_c \quad (7)$$

$$p_z^{t+1} = p_z^t + \delta v_c \quad (8)$$

with δ [m] representing the maximum translation. Then, given the new position \mathbf{p}^{t+1} , a new closest point \mathbf{p}_c^{t+1} , its distance $d_{p_c}^{t+1}$, and normal \mathbf{n}_c^{t+1} are estimated using the current point cloud. The alignment angle η is retrieved and

used to adjust to the new position while the distance is corrected as follows:

$$p_x^{t+1} = p_x^t + (d_{p_c}^t - d_{p_c}^{t+1})n_x^{t+1} \quad (9)$$

The updated position p_r^{t+1} and orientation η are sent to the position controller. During this phase, the operator is kept aware of the current alignment with a heatmap visualization of the UAV attitude with respect to the facing surface, along with the previously described dot. A visualization of the operator's camera feedback is reported in Fig. 14 while a schematic representation of the transition between states is given in Fig. 15.

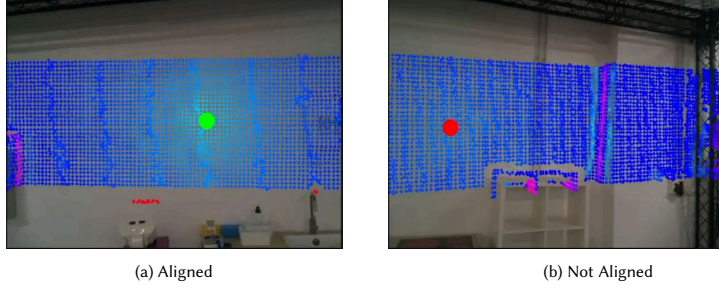


Fig. 14. Alignment visualization under two different circumstances

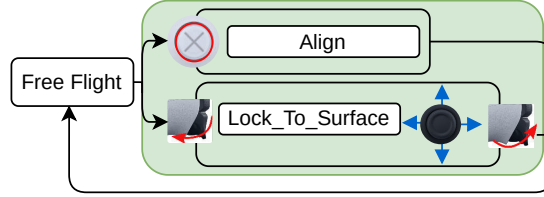


Fig. 15. Align and Lock to Surface state S transitions.

4.4 Rollercoaster

Piloting an aircraft is inherently challenging, especially when navigating in cluttered environments with limited situational awareness. To address this issue, we developed an additional plugin, which was also presented in a distinct study [14], aiming to establish an assistive navigation paradigm. In this method, the pilot controls the direction and velocity of the UAV while a planner in the background is responsible for retrieving collision-free paths and navigating towards a goal. This proposed interaction paradigm has been named Rollercoaster, and accordingly, the set of statuses is updated as follows:

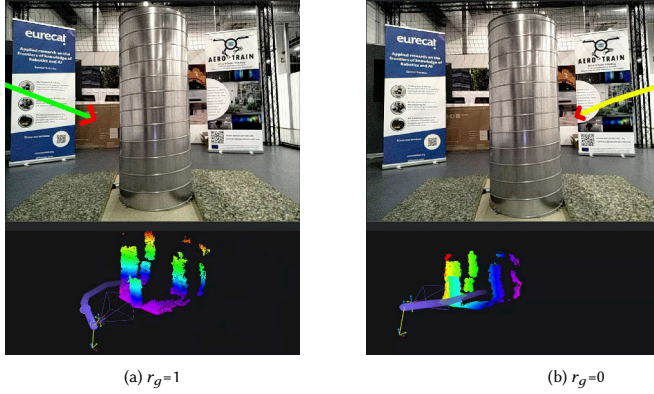
$$S = \{S \cup \{\text{Rollercoaster}\}\} \quad (10)$$

Within this plugin, the operator enables the Rollercoaster by pressing the left trigger and steers the direction of the planner with the right thumbstick, as highlighted in Fig. 16. Pressing the left trigger changes the status to $s_r =$



Fig. 16. Control interaction for Rollercoaster

Rollercoaster, while adjusting the pressure on the trigger controls the Rollercoaster gain parameter $r_g \in [0, 1]$, which regulates the velocity of the UAV along the path.

Fig. 17. Rollercoaster Visualization under different r_g

Releasing it will re-establish $s_r = \text{Free_Flight}$. Meanwhile, with the right thumbstick, the operator defines the Rollercoaster direction $r_d \in [-1, 1]$ and controls a point \mathbf{p}_d , which is at a distance k m from the current UAV position \mathbf{p}_r :

$$\tau = \arccos(r_d) \quad (11)$$

with \mathbf{p}_d :

$$p_{d_x} = p_{i_x} + k \cos(\tau) \quad (12a)$$

$$p_{d_y} = p_{i_y} + k \sin(\tau) \quad (12b)$$

$$p_{d_z} = p_{i_z} \quad (12c)$$

while the velocity is controlled with r_g , which linearly scales the maximum drone linear velocity $\mathbf{v}_{\max} = \{v_x, v_y, v_z\}$ to compute the desired velocity \mathbf{v}_n along a normalized direction vector $\mathbf{d} = \{d_x, d_y, d_z\}$ representing the direction towards

the next point in the plan:

$$\mathbf{v}_n = d\mathbf{r}_g \mathbf{v} \quad (13)$$

As previously described in Sec. 4.2, the same planner [27] along with the mapping presented in [44] are used due to their capabilities of reliably and efficiently retrieving a safe path. During navigation, the operator is kept informed with an enhanced visualization (Fig. 17), representing the retrieved path over the camera feedback where the current \mathbf{r}_g is represented as a green line over the path, and a 3D representation of the UAV moving in space.

A schematic representation of the transition between states is given in Fig. 18

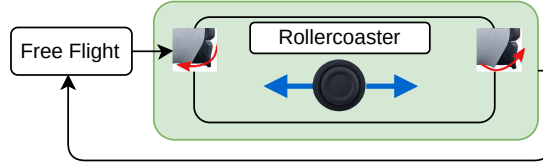


Fig. 18. Rollercoaster state S transitions.

4.5 Point to Segment to Plan

Performing visual inspections of object surfaces can be inherently complex, especially when ensuring coverage and alignment with the surfaces for accurate data collection. Therefore, we integrated the paper presented in [13] in form of plugin. This plugin aims to leverage the latest advancements in promptable segmentation models, such as [23, 20, 52], to streamline the process of defining visible surfaces or objects that require inspection. Initially, the operator defines an object for inspection by specifying points that indicate areas to include or exclude from the segmentation, performed by the segmentation model. The model then proposes the segmented area to the operator, who can confirm or refine the segmentation further. Once a satisfactory segmentation is achieved, the spatial representation of the object is retrieved and used to extract a series of waypoints covering the entire surface. These waypoints are ordered from the PathManager solving the travel salesman problem using the 2-opt [6] algorithm to obtain the shortest path. Given the complexity of the interaction paradigm, multiple statuses are added:

$$\begin{aligned} \mathcal{S} = \{ & S \cup \{ \text{Aim_To_Surface}, \\ & \text{Plan_To_Surface}, \\ & \text{Auto_Move_To_Surface}, \\ & \text{Manual_Move_To_Surface} \} \} \end{aligned} \quad (14)$$

These statuses are orchestrated with the controller interaction scheme represented in Fig. 19. Pressing the arrow down switches the operator to $s_r = \text{Aim_To_Surface}$, allowing the operator to define the points used for prompting the model through the thumbstick, with the segmented area highlighted, as shown in Fig. 20a. Pressing the arrow down again changes the status to $s_r = \text{Plan_To_Surface}$, where a spatial representation of the segmented surface is extracted, and a traversal plan at a distance q m facing the segmented surface is proposed and visualized, as in Fig. 20b.

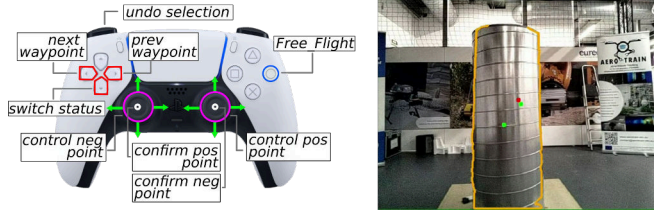


Fig. 19. Control interaction for Segment and Plan

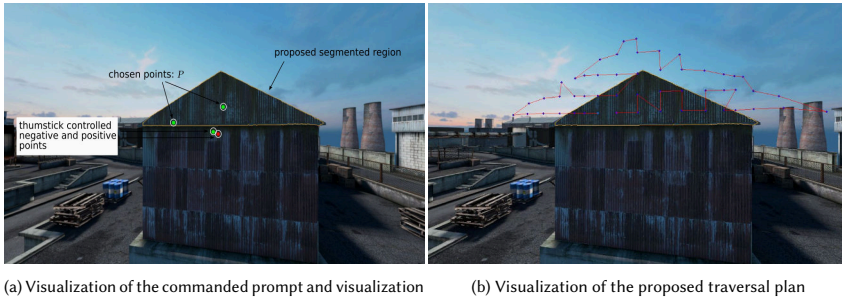


Fig. 20. Different visualization during prompting and planning

To initiate movement, the operator presses the down arrow again, transitioning the status to $s_r = \text{Auto_Move_To_Surface}$. Subsequently, the UAV autonomously follows the list of waypoints as described in Section 4.2. Throughout the operation, the operator is presented with an interface displaying the path completion (Figure 21a) and a 3D plan (Figure 21b).

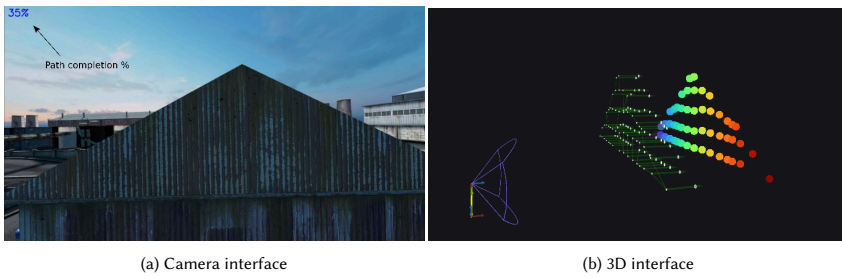


Fig. 21. Operator interfaces while moving along the plan

Since the operator may want to retain control or spend more time at a particular waypoint or inspect a certain spot again, pressing the arrow down button again changes the state to $s_r = \text{Manual_Move_To_Surface}$. At this point,

pressing the left and right arrows moves the UAV forward or backward along the list of waypoints, while pressing the circle button returns to $s_r = \text{Free_Flight}$. A clearer representation of the status changes is provided in Fig. 22.

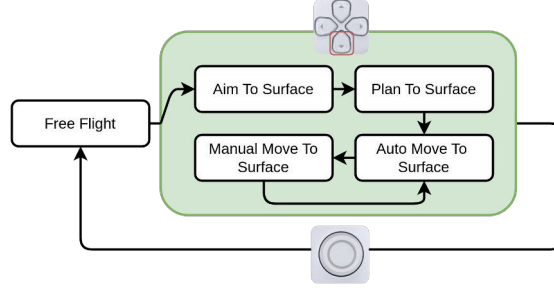


Fig. 22. This scheme represents the transition between different states (S), with the downward arrow button responsible for switching between states and a circle used to regain control

5 CONCLUSION

In conclusion, the framework presented in this paper introduces and demonstrates an innovative approach to enhance UAV teleoperation by proposing a flexible architecture capable of blending familiar teleoperation interactions with a series of shared-autonomy routines and augmented visualization. The AeroAssistant framework offers a versatile foundation capable of accommodating various features. Throughout this paper, we have demonstrated the capabilities of the framework by proposing a series of plugins that seamlessly integrate within the architecture. Each of these plugins enhances the natural and intuitive interaction with the UAV, thereby enabling even non-expert users to navigate through complex environment with ease. Future areas of interest include the integration of immersive visualization devices such as head-mounted displays, the incorporation of multi-robot systems, and the implementation of additional plugins to address various use cases. These may include search and rescue operations or aerial manipulation activities.

ACKNOWLEDGMENTS

Supported by the European Unions Horizon 2020 Research and Innovation Programme AERO-TRAIN under Grant Agreement No. 953454 and by the Catalan Government through the funding grant ACCIÓ-Eurecat (Project TRAÇA – EUTFS).

REFERENCES

- [1] Jacopo Aleotti et al. "Detection of Nuclear Sources by UAV Teleoperation Using a Visuo-Haptic Augmented Reality Interface". In: *Sensors* 17.10 (2017). ISSN: 1424-8220. DOI: [10.3390/s17102234](https://doi.org/10.3390/s17102234). URL: <https://www.mdpi.com/1424-8220/17/10/2234>.
- [2] Raffaele Brilli et al. "Monocular Reactive Collision Avoidance Based on Force Fields for Enhancing the Teleoperation of MAVs". In: *2021 20th International Conference on Advanced Robotics (ICAR)*. 2021, pp. 91–98. DOI: [10.1109/ICAR53236.2021.9659337](https://doi.org/10.1109/ICAR53236.2021.9659337).
- [3] Jonathan Cacace, Alberto Finzi, and Vincenzo Lippiello. "A mixed-initiative control system for an Aerial Service Vehicle supported by force feedback". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 1230–1235. DOI: [10.1109/IROS.2014.6942714](https://doi.org/10.1109/IROS.2014.6942714).
- [4] Linfeng Chen et al. "PinpointFly: An Egocentric Position-control Drone Interface using Mobile AR". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: [10.1145/3411764.3445110](https://doi.org/10.1145/3411764.3445110). URL: <https://doi.org/10.1145/3411764.3445110>.
- [5] Intel Corporation. *Intel® NUC*. Accessed on 2023-11-16. 2023. URL: <https://www.intel.com/content/www/us/en/developer/overview.html>.

- [6] G. A. Croes. "A Method for Solving Traveling-Salesman Problems". In: *Operations Research* 6.6 (1958), pp. 791–812. issn: 0030364X, 15265463. url: <http://www.jstor.org/stable/167074> (visited on 08/16/2023).
- [7] Martin Danelljan et al. "Adaptive Color Attributes for Real-Time Visual Tracking". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1090–1097. doi: [10.1109/CVPR.2014.143](https://doi.org/10.1109/CVPR.2014.143).
- [8] Florian Dietrich et al. "MAV tele-operation constrained on virtual surfaces for inspection of infrastructures". In: *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. Vol. 1. 2020, pp. 1519–1525. doi: [10.1109/ETFA46521.2020.9212046](https://doi.org/10.1109/ETFA46521.2020.9212046).
- [9] DJI. *DJI Mobile SDK Documentation: Component Guide - Flight Controller*. DJI. Accessed: 2024. url: <https://developer.dji.com/mobile-sdk/documentation/introduction/component-guide-flightController.html>.
- [10] Sony Interactive Entertainment. *DualSense Wireless Controller*. <https://www.playstation.com/en-us/accessories/dualsense-wireless-controller/>. 2021.
- [11] Okan Erat et al. "Drone-Augmented Human Vision: Exocentric Control for Drones Exploring Hidden Areas". In: *IEEE Transactions on Visualization and Computer Graphics* 24.4 (2018), pp. 1437–1446. doi: [10.1109/TVCG.2018.2794058](https://doi.org/10.1109/TVCG.2018.2794058).
- [12] Riccardo Franceschini, Matteo Fumagalli, and Julian Cayero Becerra. "Enhancing Human-Drone Interaction with Human-Meaningful Visual Feedback and Shared-Control Strategies". In: *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2023, pp. 1162–1167. doi: [10.1109/ICUAS57906.2023.10156190](https://doi.org/10.1109/ICUAS57906.2023.10156190).
- [13] Riccardo Franceschini, Matteo Javier Rodriguez Marquez Fumagalli, and Julian Cayero Becerra. "Point, Segment, and Inspect: Leveraging Promptable Segmentation Models for Semi-Autonomous Aerial Inspection". In: *2024 33rd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. 2024, pp. 16–23. doi: -.
- [14] Riccardo Franceschini et al. "Riding the Rollercoaster: Improving UAV Piloting Skills with Augmented Visualization and Collaborative Planning". In: *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2024, pp. 1093–1100. doi: [10.1109/ICUAS60882.2024.10556953](https://doi.org/10.1109/ICUAS60882.2024.10556953).
- [15] Boris Gromov et al. "Intuitive 3D Control of a Quadrotor in User Proximity with Pointing Gestures". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 5964–5971. doi: [10.1109/ICRA40945.2020.9196654](https://doi.org/10.1109/ICRA40945.2020.9196654).
- [16] Hooman Hedayati, Michael Walker, and Daniel Szafir. "Improving Collocated Robot Teleoperation with Augmented Reality". In: *2018 13th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2018, pp. 78–86.
- [17] Pieter Hintjens. *0MQ - The Guide*. 2011. url: <http://zguide.zeromq.org/page:all>.
- [18] Baichuan Huang et al. "Flight, Camera, Action! Using Natural Language and Mixed Reality to Control a Drone". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 6949–6956. doi: [10.1109/ICRA.2019.8794200](https://doi.org/10.1109/ICRA.2019.8794200).
- [19] Christoforos Kanellakis et al. "Towards Visual Inspection of Wind Turbines: A Case of Visual Data Acquisition Using Autonomous Aerial Robots". In: *IEEE Access* 8 (2020). Conference Name: IEEE Access, pp. 181650–181661. issn: 2169-3536. doi: [10.1109/ACCESS.2020.3028195](https://doi.org/10.1109/ACCESS.2020.3028195).
- [20] Lei Ke et al. "Segment Anything in High Quality". In: *NeurIPS*. 2023.
- [21] Leonid Keselman et al. "Intel RealSense Stereoscopic Depth Cameras". In: *CoRR* abs/1705.05548 (2017). arXiv: [1705.05548](https://arxiv.org/abs/1705.05548). url: <http://arxiv.org/abs/1705.05548>.
- [22] Dong-Hyun Kim, Yong-Guk Go, and Soo-Mi Choi. "An Aerial Mixed-Reality Environment for First-Person-View Drone Flying". In: *Applied Sciences* 10.16 (2020). issn: 2076-3417. doi: [10.3390/app10165436](https://doi.org/10.3390/app10165436). url: <https://www.mdpi.com/2076-3417/10/16/5436>.
- [23] Alexander Kirillov et al. *Segment Anything*. 2023. arXiv: [2304.02643](https://arxiv.org/abs/2304.02643) [cs.CV].
- [24] Hung M. La et al. "Mechatronic Systems Design for an Autonomous Robotic System for High-Efficiency Bridge Deck Inspection and Evaluation". In: *IEEE/ASME Transactions on Mechatronics* 18.6 (2013), pp. 1655–1664. doi: [10.1109/TMECH.2013.2279751](https://doi.org/10.1109/TMECH.2013.2279751).
- [25] LattePanda. *LattePanda 3 Delta*. <https://www.lattepanda.com/lattepanda-3-delta>. Accessed: 2024-04-04.
- [26] Chuhao Liu and Shaojie Shen. "An Augmented Reality Interaction Interface for Autonomous Drone". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 11419–11424. doi: [10.1109/IROS45743.2020.9341037](https://doi.org/10.1109/IROS45743.2020.9341037).
- [27] Sikang Liu et al. "Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments". In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1688–1695. doi: [10.1109/LRA.2017.2663526](https://doi.org/10.1109/LRA.2017.2663526).
- [28] Dylan P. Losey and Marcia K. O'Malley. "Trajectory Deformations From Physical Human–Robot Interaction". In: *IEEE Transactions on Robotics* 34.1 (2018), pp. 126–138. doi: [10.1109/TRO.2017.2765335](https://doi.org/10.1109/TRO.2017.2765335).
- [29] Steven Macenski et al. "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science Robotics* 7.66 (2022), eabm6074. doi: [10.1126/scirobotics.abm6074](https://doi.org/10.1126/scirobotics.abm6074). url: <https://www.science.org/doi/abs/10.1126/scirobotics.abm6074>.
- [30] Carlo Masone et al. "Semi-autonomous trajectory generation for mobile robots with integral haptic shared control". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 6468–6475. doi: [10.1109/ICRA.2014.6907814](https://doi.org/10.1109/ICRA.2014.6907814).
- [31] Carlo Masone et al. "Shared planning and control for mobile robots with integral haptic feedback". In: *The International Journal of Robotics Research* 37.11 (2018), pp. 1395–1420. doi: [10.1177/0278364918802006](https://doi.org/10.1177/0278364918802006). eprint: <https://doi.org/10.1177/0278364918802006>. url: <https://doi.org/10.1177/0278364918802006>.
- [32] Lorenz Meier, Dominik Honegger, and Marc Pollefeys. "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms". In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 6235–6240. doi: [10.1109/ICRA.2015.7140074](https://doi.org/10.1109/ICRA.2015.7140074).
- [33] ModalAI. *VOXL2: A Powerful Companion Computer for Drones*. <https://www.modalai.com/products/voxl-2?variant=39914779836467>. Accessed 2024-02-08.

- [34] Marcin Odelga, Paolo Stegagno, and Heinrich H. Bühlhoff. "Obstacle detection, tracking and avoidance for a teleoperated UAV". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 2984–2990. doi: [10.1109/ICRA.2016.7487464](https://doi.org/10.1109/ICRA.2016.7487464).
- [35] Marcin Odelga et al. "A Self-contained Teleoperated Quadrotor: On-Board State-Estimation and Indoor Obstacle Avoidance". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 7840–7847. doi: [10.1109/ICRA.2018.8463185](https://doi.org/10.1109/ICRA.2018.8463185).
- [36] Fatma Outay, Hanan Abdullah Mengash, and Muhammad Adnan. "Applications of unmanned aerial vehicle (UAV) in road safety, traffic and highway infrastructure management: Recent advances and challenges". In: *Transportation Research Part A: Policy and Practice* 141 (Nov. 2020), pp. 116–129. ISSN: 09658564. doi: [10.1016/j.tra.2020.09.018](https://doi.org/10.1016/j.tra.2020.09.018). URL: <https://linkinghub.elsevier.com/retrieve/pii/S096585642030728X> (visited on 07/06/2021).
- [37] F.J. Perez-Grau et al. "Semi-autonomous teleoperation of UAVs in search and rescue scenarios". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2017, pp. 1066–1074. doi: [10.1109/ICUAS.2017.7991349](https://doi.org/10.1109/ICUAS.2017.7991349).
- [38] Qualcomm Developer Network. *Machine Vision SDK*. Accessed: 2024-05-27. 2024. URL: <https://developer.qualcomm.com/software/machine-vision-sdk>.
- [39] Morgan Quigley et al. "ROS: An open-source robot operating system". In: *Workshops at the IEEE International Conference on Robotics and Automation*. 2009.
- [40] Fran Real et al. "Unmanned aerial vehicle abstraction layer: An abstraction layer to operate unmanned aerial vehicles". In: *International Journal of Advanced Robotic Systems* 17.4 (2020), pp. 1–13. doi: [10.1177/1729881420925011](https://doi.org/10.1177/1729881420925011). URL: <https://doi.org/10.1177/1729881420925011>.
- [41] P. J. Sanchez-Cuevas, G. Heredia, and A. Ollero. "Multirotor UAS for bridge inspection by contact using the ceiling effect". In: *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. 2017 International Conference on Unmanned Aircraft Systems (ICUAS). June 2017, pp. 767–774. doi: [10.1109/ICUAS.2017.7991412](https://doi.org/10.1109/ICUAS.2017.7991412).
- [42] Oscar Bowen Schofield, Kasper Høj Lorenzen, and Emad Ebeid. "Cloud to Cable: A Drone Framework for Autonomous Power line Inspection". In: *2020 23rd Euromicro Conference on Digital System Design (DSD)*. 2020, pp. 503–509. doi: [10.1109/DSD51259.2020.00085](https://doi.org/10.1109/DSD51259.2020.00085).
- [43] Ryo Suzuki et al. "Augmented Reality and Robotics: A Survey and Taxonomy for AR-Enhanced Human-Robot Interaction and Robotic Interfaces". In: *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI '22. New Orleans, LA, USA: Association for Computing Machinery, 2022. ISBN: 9781450391573. doi: [10.1145/3491102.3517719](https://doi.org/10.1145/3491102.3517719). URL: <https://doi.org/10.1145/3491102.3517719>.
- [44] Jesus Tordesillas et al. "Real-Time Planning with Multi-Fidelity Models for Agile Flights in Unknown Environments". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 725–731. doi: [10.1109/ICRA.2019.8794248](https://doi.org/10.1109/ICRA.2019.8794248).
- [45] Diego Vaquero-Melchor and Ana M. Bernardos. "Alternative interaction techniques for drone-based mission definition: from desktop UI to wearable AR". In: *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia*. MUM '19. Pisa, Italy: Association for Computing Machinery, 2019. ISBN: 9781450376242. doi: [10.1145/3365610.3368420](https://doi.org/10.1145/3365610.3368420). URL: <https://doi.org/10.1145/3365610.3368420>.
- [46] Karl D. von Ellenrieder et al. "Shared human-robot path following control of an unmanned ground vehicle". In: *Mechatronics* 83 (2022), p. 102750. ISSN: 0957-4158. doi: <https://doi.org/10.1016/j.mechatronics.2022.102750>. URL: <https://www.sciencedirect.com/science/article/pii/S0957415822000083>.
- [47] Michael E. Walker, Hooman Hedayati, and Daniel Szafir. "Robot Teleoperation with Augmented Reality Virtual Surrogates". In: *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. 2019, pp. 202–210. doi: [10.1109/HRI.2019.8673306](https://doi.org/10.1109/HRI.2019.8673306).
- [48] Qianhao Wang et al. "GPA-Teleoperation: Gaze Enhanced Perception-Aware Safe Assistive Aerial Teleoperation". In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 5631–5638. doi: [10.1109/LRA.2022.3153898](https://doi.org/10.1109/LRA.2022.3153898).
- [49] Mohammad Kassem Zein et al. "Deep Learning and Mixed Reality to Autocomplete Teleoperation". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 4523–4529. doi: [10.1109/ICRA48506.2021.9560887](https://doi.org/10.1109/ICRA48506.2021.9560887).
- [50] Mohammad Kassem Zein et al. "Enhanced Teleoperation Using Autocomplete". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 9178–9184. doi: [10.1109/ICRA40945.2020.9197140](https://doi.org/10.1109/ICRA40945.2020.9197140).
- [51] Dawei Zhang, Guang Yang, and Rebecca P. Khurshid. "Haptic Teleoperation of UAVs Through Control Barrier Functions". In: *IEEE Transactions on Haptics* 13.1 (2020), pp. 109–115. doi: [10.1109/TOH.2020.2966485](https://doi.org/10.1109/TOH.2020.2966485).
- [52] Xu Zhao et al. *Fast Segment Anything*. 2023. arXiv: 2306.12156 [cs. CV].

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009

Bibliography

- [1] OECD, *Infrastructure to 2030*. 2006.
- [2] F. Outay, H. A. Mengash, and M. Adnan, “Applications of unmanned aerial vehicle (UAV) in road safety, traffic and highway infrastructure management: Recent advances and challenges,” vol. 141, pp. 116–129.
- [3] H. M. La, R. S. Lim, B. B. Basily, N. Gucunski, J. Yi, A. Maher, F. A. Romero, and H. Parvardeh, “Mechatronic systems design for an autonomous robotic system for high-efficiency bridge deck inspection and evaluation,” *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1655–1664, 2013.
- [4] P. J. Sanchez-Cuevas, G. Heredia, and A. Ollero, “Multirotor UAS for bridge inspection by contact using the ceiling effect,” in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 767–774.
- [5] O. B. Schofield, K. H. Lorenzen, and E. Ebeid, “Cloud to cable: A drone framework for autonomous power line inspection,” in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, pp. 503–509, 2020.
- [6] DJI, “Solar panel dji,”
- [7] C. Kanellakis, E. Fresk, S. S. Mansouri, D. Kominiak, and G. Nikolakopoulos, “Towards visual inspection of wind turbines: A case of visual data acquisition using autonomous aerial robots,” vol. 8, pp. 181650–181661. Conference Name: IEEE Access.
- [8] S. Ljungblad, Y. Man, M. A. Baytaş, M. Gamboa, M. Obaid, and M. Fjeld, “What matters in professional drone pilots’ practice? an interview study to understand the complexity of their work and inform human-drone interaction research,” in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–16, ACM.
- [9] K. Okamura and S. Yamada, “Calibrating trust in human-drone cooperative navigation,” in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 1274–1279. ISSN: 1944-9437.
- [10] J. M. Beer, A. D. Fisk, and W. A. Rogers, “Toward a framework for levels of robot autonomy in human-robot interaction,” vol. 3, no. 2, p. 74.

- [11] C. Brooks and D. Szafr, "Visualization of intended assistance for acceptance of shared control," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11425–11430, 2020.
- [12] I. D. Foundation, "Beyond ar vs. vr: What is the difference between ar vs. mr vs. vr vs. xr?," <https://www.interaction-design.org/literature/article/beyond-ar-vs-vr-what-is-the-difference-between-ar-vs-mr-vs-vr-vs-xr>, 2022.
- [13] R. Suzuki, A. Karim, T. Xia, H. Hedayati, and N. Marquardt, "Augmented reality and robotics: A survey and taxonomy for ar-enhanced human-robot interaction and robotic interfaces," in *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, CHI '22, (New York, NY, USA), Association for Computing Machinery, 2022.
- [14] M. Odelga, P. Stegagno, N. Kochanek, and H. H. Bühlhoff, "A self-contained teleoperated quadrotor: On-board state-estimation and indoor obstacle avoidance," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7840–7847, 2018.
- [15] M. Odelga, P. Stegagno, and H. H. Bühlhoff, "Obstacle detection, tracking and avoidance for a teleoperated uav," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2984–2990, 2016.
- [16] R. Brilli, M. Pozzi, F. Giorgetti, M. L. Fravolini, P. Valigi, D. Prattichizzo, and G. Costante, "Monocular reactive collision avoidance based on force fields for enhancing the teleoperation of mavs," in *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 91–98, 2021.
- [17] F. Dietrich, J. Marzat, M. Sanfourche, S. Bertrand, A. Bernard-Brunel, and A. Eudes, "Mav tele-operation constrained on virtual surfaces for inspection of infrastructures," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, pp. 1519–1525, 2020.
- [18] C. Liu and S. Shen, "An augmented reality interaction interface for autonomous drone," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 11419–11424, 2020.
- [19] M. E. Walker, H. Hedayati, and D. Szafr, "Robot teleoperation with augmented reality virtual surrogates," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 202–210, 2019.
- [20] L. Chen, K. Takashima, K. Fujita, and Y. Kitamura, "Pinpointfly: An ego-centric position-control drone interface using mobile ar," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, (New York, NY, USA), Association for Computing Machinery, 2021.

- [21] D. Vaquero-Melchor and A. M. Bernardos, “Alternative interaction techniques for drone-based mission definition: from desktop ui to wearable ar,” in *Proceedings of the 18th International Conference on Mobile and Ubiquitous Multimedia*, MUM '19, (New York, NY, USA), Association for Computing Machinery, 2019.
- [22] Y.-A. Chen, T.-Y. Wu, T. Chang, J. Y. Liu, Y.-C. Hsieh, L. Y. Hsu, M.-W. Hsu, P. Taele, N.-H. Yu, and M. Y. Chen, “ARPilot: designing and investigating AR shooting interfaces on mobile devices for drone videography,” in *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pp. 1–8, ACM.
- [23] B. Huang, D. Bayazit, D. Ullman, N. Gopalan, and S. Tellex, “Flight, camera, action! using natural language and mixed reality to control a drone,” in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6949–6956, 2019.
- [24] M. Riechmann, A. Kirsch, and M. Koenig, “Augmented reality for interactive path planning in 3d,” in *2023 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 561–566, 2023.
- [25] J. Cacace, A. Finzi, and V. Lippiello, “A mixed-initiative control system for an aerial service vehicle supported by force feedback,” in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1230–1235, 2014.
- [26] C. Masone, P. R. Giordano, H. H. Bühlhoff, and A. Franchi, “Semi-autonomous trajectory generation for mobile robots with integral haptic shared control,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6468–6475, 2014.
- [27] D. P. Losey and M. K. O'Malley, “Trajectory deformations from physical human-robot interaction,” *IEEE Transactions on Robotics*, vol. 34, no. 1, pp. 126–138, 2018.
- [28] K. D. von Ellenrieder, S. C. Licht, R. Belotti, and H. C. Henninger, “Shared human-robot path following control of an unmanned ground vehicle,” *Mechatronics*, vol. 83, p. 102750, 2022.
- [29] C. Masone, M. Mohammadi, P. R. Giordano, and A. Franchi, “Shared planning and control for mobile robots with integral haptic feedback,” *The International Journal of Robotics Research*, vol. 37, no. 11, pp. 1395–1420, 2018.
- [30] D. Zhang, G. Yang, and R. P. Khurshid, “Haptic teleoperation of uavs through control barrier functions,” *IEEE Transactions on Haptics*, vol. 13, no. 1, pp. 109–115, 2020.
- [31] M. K. Zein, A. Sidaoui, D. Asmar, and I. H. Elhajj, “Enhanced teleoperation using autocomplete,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9178–9184, 2020.

- [32] M. K. Zein, M. Al Aawar, D. Asmar, and I. H. Elhajj, "Deep learning and mixed reality to autocomplete teleoperation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4523–4529, 2021.
- [33] B. Ibrahim, M. H. Hussein, I. H. Elhajj, and D. Asmar, "Autocomplete of 3d motions for uav teleoperation," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7825–7831, 2023.
- [34] B. Gromov, J. Guzzi, L. M. Gambardella, and A. Giusti, "Intuitive 3d control of a quadrotor in user proximity with pointing gestures," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5964–5971, 2020.
- [35] A. Menshchikov, D. Ermilov, I. Dranitsky, L. Kupchenko, M. Panov, M. Fedorov, and A. Somov, "Data-driven body-machine interface for drone intuitive control through voice and gestures," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, pp. 5602–5609. ISSN: 2577-1647.
- [36] E. Peshkova, M. Hitz, and B. Kaufmann, "Natural interaction techniques for an unmanned aerial vehicle system," vol. 16, no. 1, pp. 34–42. Conference Name: IEEE Pervasive Computing.
- [37] J. DelPreto and D. Rus, "Plug-and-play gesture control using muscle and motion sensors," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, pp. 439–448, ACM.
- [38] "The hover." <https://thehover.com/>. Accessed: April 9, 2024.
- [39] "Pixy." <https://www.pixy.com/>. Accessed: April 9, 2024.
- [40] Q. Wang, B. He, Z. Xun, C. Xu, and F. Gao, "Gpa-teleoperation: Gaze enhanced perception-aware safe assistive aerial teleoperation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 5631–5638, 2022.
- [41] O. Erat, W. A. Isop, D. Kalkofen, and D. Schmalstieg, "Drone-augmented human vision: Exocentric control for drones exploring hidden areas," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 4, pp. 1437–1446, 2018.
- [42] H. Hedayati, M. Walker, and D. Szafir, "Improving collocated robot teleoperation with augmented reality," in *2018 13th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 78–86, 2018.
- [43] D.-H. Kim, Y.-G. Go, and S.-M. Choi, "An aerial mixed-reality environment for first-person-view drone flying," *Applied Sciences*, vol. 10, no. 16, 2020.

- [44] M. Inoue, K. Takashima, K. Fujita, and Y. Kitamura, “Birdviewer: Surroundings-aware remote drone piloting using an augmented third-person perspective,” in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, (New York, NY, USA), Association for Computing Machinery, 2023.
- [45] M. Allenspach, T. Kötter, R. Bähnemann, M. Tognon, and R. Siegwart, “Design and evaluation of a mixed reality-based human-robot interface for teleoperation of omnidirectional aerial vehicles,” in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1168–1174, 2023.
- [46] M. Allenspach, S. Laasch, N. Lawrance, M. Tognon, and R. Siegwart, “Mixed reality human-robot interface to generate and visualize 6dof trajectories: Application to omnidirectional aerial vehicles,” in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 395–400, 2023.
- [47] G. A. Yashin, D. Trinitatova, R. T. Agishev, R. Ibrahimov, and D. Tsetserukou, “AeroVR: Virtual reality-based teleoperation with tactile feedback for aerial manipulation,”
- [48] D. Kim and P. Y. Oh, “Aerial manipulation using a human-embodied drone interface,” in *2022 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*, pp. 1–7, 2022.
- [49] “DJI goggles 2.” <https://www.dji.com/uk/goggles-2>. Accessed: April 9, 2024.
- [50] R. Franceschini, M. Fumagalli, and J. C. Becerra, “Learn to efficiently exploit cost maps by combining rrt* with reinforcement learning,” in *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 251–256, 2022.
- [51] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
- [52] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *CoRR*, vol. abs/1707.06347, 2017.
- [53] D. Devaurs, T. Siméon, and J. Cortés, “Optimal path planning in complex cost spaces with sampling-based algorithms,” *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 415–424, 2016.
- [54] R. Franceschini, M. Fumagalli, and J. C. Becerra, “Enhancing human-drone interaction with human-meaningful visual feedback and shared-control strategies,” in *2023 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1162–1167, 2023.

- [55] R. Franceschini, M. Fumagalli, and J. C. Becerra, "Riding the rollercoaster: easing UAV piloting experience with XR and continuous planning," in *XR-ROB 2023 - Second International Workshop on "Horizons of an Extended Robotics Reality" @ IEEE/RSJ IROS 2023*, 2023.
- [56] R. Franceschini, J. R. Marquez, M. Fumagalli, and J. C. Becerra, "Riding the rollercoaster: Improving uav piloting skills with augmented visualization and collaborative planning," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 1093–1100, 2024.
- [57] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-d complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.
- [58] J. Tordesillas, B. T. Lopez, J. Carter, J. Ware, and J. P. How, "Real-time planning with multi-fidelity models for agile flights in unknown environments," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 725–731, 2019.
- [59] S. I. Entertainment, "Dualsense wireless controller." <https://www.playstation.com/en-us/accessories/dualsense-wireless-controller/>, 2021.
- [60] S. G. Hart and L. E. Staveland, "Development of nasa-tlx (task load index): Results of empirical and theoretical research," *Advances in psychology*, vol. 52, pp. 139–183, 1988.
- [61] ModalAI, "VOXL2: A Powerful Companion Computer for Drones." <https://www.modalai.com/products/vox1-2?variant=39914779836467>, Accessed 2024-02-08.
- [62] 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.
- [63] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel real-sense stereoscopic depth cameras," *CoRR*, vol. abs/1705.05548, 2017.
- [64] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment anything," 2023.
- [65] G. A. Croes, "A method for solving traveling-salesman problems," *Operations Research*, vol. 6, no. 6, pp. 791–812, 1958.

- [66] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, “Flightmare: A flexible quadrotor simulator,” in *Conference on Robot Learning*, 2020.
- [67] L. Ke, M. Ye, M. Danelljan, Y. Liu, Y.-W. Tai, C.-K. Tang, and F. Yu, “Segment anything in high quality,” in *NeurIPS*, 2023.
- [68] X. Zhao, W. Ding, Y. An, Y. Du, T. Yu, M. Li, M. Tang, and J. Wang, “Fast segment anything,” 2023.
- [69] R. Franceschini, M. Fumagalli, and J. C. Becerra, “Aeroassistant: a modern and flexible teleoperation framework,” (New York, NY, USA), Association for Computing Machinery, 2024.
- [70] F. Real, A. Torres-González, P. R. Soria, J. Capitán, and A. Ollero, “Unmanned aerial vehicle abstraction layer: An abstraction layer to operate unmanned aerial vehicles,” *International Journal of Advanced Robotic Systems*, vol. 17, no. 4, pp. 1–13, 2020.
- [71] LattePanda, “LattePanda 3 Delta.” <https://www.lattepanda.com/lattepanda-3-delta>, Accessed: 2024-04-04.
- [72] I. Corporation, “Intel® nuc,” 2023. Accessed on 2023-11-16.
- [73] Q. Zhou, J. Park, and V. Koltun, “Open3d: A modern library for 3d data processing,” *CoRR*, vol. abs/1801.09847, 2018.
- [74] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.
- [75] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “ROS: An open-source robot operating system,” in *Workshops at the IEEE International Conference on Robotics and Automation*, 2009.
- [76] 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.
- [77] P. Hintjens, “Omq - the guide,” 2011.
- [78] M. Danelljan, F. S. Khan, M. Felsberg, and J. Van De Weijer, “Adaptive color attributes for real-time visual tracking,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1090–1097, 2014.
- [79] R. Franceschini, M. Fumagalli, and J. C. Becerra, “Point, segment, and inspect: Leveraging promptable segmentation models for semi-autonomous aerial inspection,” in *2024 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pp. 16–23, 2024.

- [80] Unity Technologies, “Unity AR Foundation Manual.” <https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@6.0/manual/index.html>, 2024.
- [81] PTC, “Vuforia Engine Documentation.” <https://www.ptc.com/en/products/vuforia/vuforia-engine/>, 2024.
- [82] Google, “Google ARCore Documentation.” <https://developers.google.com/ar>, 2024.
- [83] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 ed., 2004.
- [84] C. Staff, “Camera models and their applications,” 2023. Accessed: 2024-05-14.
- [85] “NVIDIA Jetson Orin.” <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/>, Accessed: 2024. Accessed on: 2024-04-23.
- [86] “Raspberry Pi 4 Model B.” <https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>, Accessed: 2024. Accessed on: 2024-04-23.
- [87] NVIDIA, P. Vingelmann, and F. H. Fitzek, “Cuda, release: 10.2.89,” 2020.
- [88] “Qualcomm QRB5165.” <https://www.qualcomm.com/products/internet-of-things/industrial/industrial-automation/qrb5165#Overview>, Accessed: 2024. Accessed on: 2024-04-23.
- [89] G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” *arXiv preprint arXiv:1503.02531*, 2015.
- [90] K. Wu, J. Zhang, H. Peng, M. Liu, B. Xiao, J. Fu, and L. Yuan, “Tinyvit: Fast pretraining distillation for small vision transformers,” 2022.
- [91] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, “Yolact: Real-time instance segmentation,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.
- [92] C. White, M. Safari, R. Sukthanker, B. Ru, T. Elsken, A. Zela, D. Dey, and F. Hutter, “Neural architecture search: Insights from 1000 papers,” *arXiv preprint arXiv:2301.08727*, 2023.
- [93] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *CoRR*, vol. abs/1611.01578, 2016.
- [94] E. Cereda, L. Crupi, M. Risso, A. Burrello, L. Benini, A. Giusti, D. Jahier Pagliari, and D. Palossi, “Deep neural network architecture search for accurate visual pose estimation aboard nano-uavs,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6065–6071, 2023.

- [95] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. G. Howard, H. Adam, and D. Kalenichenko, “Quantization and training of neural networks for efficient integer-arithmetic-only inference,” *CoRR*, vol. abs/1712.05877, 2017.
- [96] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “Pytorch: An imperative style, high-performance deep learning library,” *CoRR*, vol. abs/1912.01703, 2019.
- [97] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zhang, “Tensorflow: A system for large-scale machine learning,” *CoRR*, vol. abs/1605.08695, 2016.
- [98] J. Bai, F. Lu, K. Zhang, *et al.*, “Onnx: Open neural network exchange.” <https://github.com/onnx/onnx>, 2019.
- [99] “PyTorch ExecuTorch.” <https://pytorch.org/executorch-overview>, Accessed: 2024. Accessed on: 2024-04-23.
- [100] “TensorFlow Lite.” <https://www.tensorflow.org/lite>, Accessed: 2024. Accessed on: 2024-04-23.
- [101] PINTO0309, “onnx2tf: Convert ONNX to TensorFlow.” <https://github.com/PINTO0309/onnx2tf>, 2024.
- [102] T. D. Science, “Non-maximum suppression (nms),” Accessed: 2024. Accessed on: 2024-04-23.
- [103] Livox Technology Co., Ltd., “Livox mid-360,” Accessed 2024. Accessed on 13th May 2024.
- [104] W. Xu and F. Zhang, “Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021.
- [105] Meta Platforms, Inc., “Quest 3.” <https://www.meta.com/quest/quest-3/>, 2024.
- [106] Apple Inc., “Apple Vision Pro.” <https://www.apple.com/apple-vision-pro/>, 2024.
- [107] Unity Technologies, “Unity,” 2023. Game development platform.
- [108] M. Divband Soorati, J. Clark, J. Ghofrani, D. Tarapore, and S. D. Ramchurn, “Designing a user-centered interaction interface for human–swarm teaming,” *Drones*, vol. 5, no. 4, 2021.

- [109] M. Macchini, L. De Matteis, F. Schiano, and D. Floreano, "Personalized human-swarm interaction through hand motion," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8341–8348, 2021.
- [110] M. Pavliv, F. Schiano, C. Reardon, D. Floreano, and G. Loianno, "Tracking and relative localization of drone swarms with a vision-based headset," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1455–1462, 2021.
- [111] Pico Interactive, "Pico XR." <https://www.picoxr.com/global>, 2024.
- [112] "Varjo XR-4." <https://varjo.com/products/xr-4/>, Accessed: 2024. Accessed on: 2024-04-23.
- [113] HTC Corporation, "VIVE." <https://www.vive.com/us/>, 2024.
- [114] A. Ollero, M. Tognon, A. Suarez, D. Lee, and A. Franchi, "Past, present, and future of aerial robotic manipulators," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 626–645, 2022.
- [115] R. Dautzenberg, T. Küster, T. Mathis, Y. Roth, C. Steinauer, G. Käppeli, J. Santen, A. Arranhado, F. Biffar, T. Kötter, C. Lanegger, M. Allenspach, R. Siegwart, and R. Bähnmann, "A perching and tilting aerial robot for precise and versatile power tool work on vertical walls," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1094–1101, 2023.
- [116] J. Cacace, G. A. Fontanelli, and V. Lippiello, "A novel hybrid aerial-ground manipulator for pipeline inspection tasks," in *2021 Aerial Robotic Systems Physically Interacting with the Environment (AIRPHARO)*, pp. 1–6, 2021.
- [117] A. Suarez, G. Heredia, and A. Ollero, "Design of an anthropomorphic, compliant, and lightweight dual arm for aerial manipulation," *IEEE Access*, vol. 6, pp. 29173–29189, 2018.
- [118] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Asbell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," 2022.
- [119] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.
- [120] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed, "Mistral 7b," 2023.

- [121] G. Team, R. Anil, S. Borgeaud, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, K. Millican, D. Silver, M. Johnson, I. Antonoglou, J. Schrittwieser, A. Glaese, J. Chen, E. Pitler, T. Lillicrap, A. Lazaridou, O. Firat, J. Molloy, M. Isard, P. R. Barham, T. Hennigan, B. Lee, F. Viola, M. Reynolds, Y. Xu, R. Doherty, E. Collins, C. Meyer, E. Rutherford, E. Moreira, K. Ayoub, M. Goel, J. Krawczyk, C. Du, E. Chi, H.-T. Cheng, E. Ni, P. Shah, P. Kane, B. Chan, M. Faruqui, A. Severyn, H. Lin, Y. Li, Y. Cheng, A. Ittycheriah, M. Mahdiah, M. Chen, P. Sun, D. Tran, S. Bagri, B. Lakshminarayanan, J. Liu, A. Orban, F. Güra, H. Zhou, X. Song, A. Boffy, H. Ganapathy, S. Zheng, H. Choe, Ágoston Weisz, T. Zhu, Y. Lu, S. Gopal, J. Kahn, M. Kula, J. Pitman, R. Shah, E. Taropa, M. A. Merey, M. Baeuml, Z. Chen, L. E. Shafey, Y. Zhang, O. Sercinoglu, G. Tucker, E. Piqueras, M. Krikun, I. Barr, N. Savinov, I. Danihelka, B. Roelofs, A. White, A. Andreassen, T. von Glehn, L. Yagati, M. Kazemi, L. Gonzalez, M. Khalman, J. Sygnowski, A. Frechette, C. Smith, L. Culp, L. Proleev, Y. Luan, X. Chen, J. Lottes, N. Schucher, F. Lebron, A. Rrustemi, N. Clay, P. Crone, T. Kocisky, J. Zhao, B. Perz, D. Yu, H. Howard, A. Bloniarz, J. W. Rae, H. Lu, L. Sifre, M. Maggioni, F. Alcober, D. Garrette, M. Barnes, S. Thakoor, J. Austin, G. Barth-Maron, W. Wong, R. Joshi, R. Chaabouni, D. Fatiha, A. Ahuja, G. S. Tomar, E. Senter, M. Chadwick, I. Kornakov, N. Attaluri, I. Iturrate, R. Liu, Y. Li, S. Cogan, J. Chen, C. Jia, C. Gu, Q. Zhang, J. Grimsstad, A. J. Hartman, X. Garcia, T. S. Pillai, J. Devlin, M. Laskin, D. de Las Casas, D. Valter, C. Tao, L. Blanco, A. P. Badia, D. Reitter, M. Chen, J. Brennan, C. Rivera, S. Brin, S. Iqbal, G. Surita, J. Labanowski, A. Rao, S. Winkler, E. Parisotto, Y. Gu, K. Olszewska, R. Addanki, A. Miech, A. Louis, D. Teplyashin, G. Brown, E. Catt, J. Balaguer, J. Xiang, P. Wang, Z. Ashwood, A. Briukhov, A. Webson, S. Ganapathy, S. Sanghavi, A. Kannan, M.-W. Chang, A. Stjerngren, J. Djolonga, Y. Sun, A. Bapna, M. Aitchison, P. Pejman, H. Michalewski, T. Yu, C. Wang, J. Love, J. Ahn, D. Bloxwich, K. Han, P. Humphreys, T. Sellam, J. Bradbury, V. Godbole, S. Samangoeei, B. Damoc, A. Kaskasoli, S. M. R. Arnold, V. Vasudevan, S. Agrawal, J. Riesa, D. Lepikhin, R. Tanburn, S. Srinivasan, H. Lim, S. Hodgkinson, P. Shyam, J. Ferret, S. Hand, A. Garg, T. L. Paine, J. Li, Y. Li, M. Giang, A. Neitz, Z. Abbas, S. York, M. Reid, E. Cole, A. Chowdhery, D. Das, D. Rogozińska, V. Nikolaev, P. Sprechmann, Z. Nado, L. Zilka, F. Prost, L. He, M. Monteiro, G. Mishra, C. Welty, J. Newlan, D. Jia, M. Allamanis, C. H. Hu, R. de Liedekerke, J. Gilmer, C. Saroufim, S. Rijhwani, S. Hou, D. Shrivastava, A. Baddepudi, A. Goldin, A. Ozturk, A. Cassirer, Y. Xu, D. Sohn, D. Sachan, R. K. Amplayo, C. Swanson, D. Petrova, S. Narayan, A. Guez, S. Brahma, J. Landon, M. Patel, R. Zhao, K. Villela, L. Wang, W. Jia, M. Rahtz, M. Giménez, L. Yeung, J. Keeling, P. Georgiev, D. Mincu, B. Wu, S. Haykal, R. Saputro, K. Vodrahalli, J. Qin, Z. Cankara, A. Sharma, N. Fernando, W. Hawkins, B. Neyshabur, S. Kim, A. Hutter, P. Agrawal, A. Castro-Ros, G. van den Driessche, T. Wang, F. Yang, S. yiin Chang, P. Komarek, R. McIlroy, M. Lučić, G. Zhang, W. Farhan, M. Sharman, P. Natsev,

P. Michel, Y. Bansal, S. Qiao, K. Cao, S. Shakeri, C. Butterfield, J. Chung, P. K. Rubenstein, S. Agrawal, A. Mensch, K. Soparkar, K. Lenc, T. Chung, A. Pope, L. Maggiore, J. Kay, P. Jhakra, S. Wang, J. Maynez, M. Phuong, T. Tobin, A. Tacchetti, M. Trebacz, K. Robinson, Y. Katariya, S. Riedel, P. Bailey, K. Xiao, N. Ghelani, L. Aroyo, A. Slone, N. Houlsby, X. Xiong, Z. Yang, E. Gribovskaya, J. Adler, M. Wirth, L. Lee, M. Li, T. Kagohara, J. Pavagadhi, S. Bridgers, A. Bortsova, S. Ghemawat, Z. Ahmed, T. Liu, R. Powell, V. Bolina, M. Iinuma, P. Zablotskaia, J. Besley, D.-W. Chung, T. Dozat, R. Comanescu, X. Si, J. Greer, G. Su, M. Polacek, R. L. Kaufman, S. Tokumine, H. Hu, E. Buchatskaya, Y. Miao, M. Elhawaty, A. Siddhant, N. Tomasev, J. Xing, C. Greer, H. Miller, S. Ashraf, A. Roy, Z. Zhang, A. Ma, A. Filos, M. Besta, R. Blevins, T. Klimenko, C.-K. Yeh, S. Changpinyo, J. Mu, O. Chang, M. Pajarskas, C. Muir, V. Cohen, C. L. Lan, K. Haridasan, A. Marathe, S. Hansen, S. Douglas, R. Samuel, M. Wang, S. Austin, C. Lan, J. Jiang, J. Chiu, J. A. Lorenzo, L. L. Sjöstrand, S. Cevey, Z. Gleicher, T. Avrahami, A. Boral, H. Srinivasan, V. Selo, R. May, K. Aisopos, L. Hussenot, L. B. Soares, K. Baumli, M. B. Chang, A. Recasens, B. Caine, A. Pritzel, F. Pavetic, F. Pardo, A. Gergely, J. Frye, V. Ramasesh, D. Horgan, K. Badola, N. Kassner, S. Roy, E. Dyer, V. C. Campos, A. Tomala, Y. Tang, D. E. Badawy, E. White, B. Mustafa, O. Lang, A. Jindal, S. Vikram, Z. Gong, S. Caelles, R. Hemsley, G. Thornton, F. Feng, W. Stokowiec, C. Zheng, P. Thacker, Çağlar Ünlü, Z. Zhang, M. Saleh, J. Svensson, M. Bileschi, P. Patil, A. Anand, R. Ring, K. Tsihlias, A. Vezer, M. Selvi, T. Shevlane, M. Rodriguez, T. Kwiatkowski, S. Daruki, K. Rong, A. Dafoe, N. FitzGerald, K. Gu-Lemberg, M. Khan, L. A. Hendricks, M. Pellat, V. Feinberg, J. Cobon-Kerr, T. Sainath, M. Rauh, S. H. Hashemi, R. Ives, Y. Hasson, E. Noland, Y. Cao, N. Byrd, L. Hou, Q. Wang, T. Sottiaux, M. Paganini, J.-B. Lespiau, A. Moufarek, S. Hassan, K. Shivakumar, J. van Amersfoort, A. Mandhane, P. Joshi, A. Goyal, M. Tung, A. Brock, H. Sheahan, V. Misra, C. Li, N. Rakićević, M. Dehghani, F. Liu, S. Mittal, J. Oh, S. Noury, E. Sezener, F. Huot, M. Lamm, N. D. Cao, C. Chen, S. Mudgal, R. Stella, K. Brooks, G. Vasudevan, C. Liu, M. Chain, N. Melinkeri, A. Cohen, V. Wang, K. Seymore, S. Zubkov, R. Goel, S. Yue, S. Krishnakumaran, B. Albert, N. Hurley, M. Sano, A. Mohananey, J. Joughin, E. Filonov, T. Kępa, Y. Eldawy, J. Lim, R. Rishi, S. Badiezadegan, T. Bos, J. Chang, S. Jain, S. G. S. Padmanabhan, S. Puttagunta, K. Krishna, L. Baker, N. Kalb, V. Bedapudi, A. Kurzkro, S. Lei, A. Yu, O. Litvin, X. Zhou, Z. Wu, S. Sobell, A. Siciliano, A. Papir, R. Neale, J. Bragagnolo, T. Toor, T. Chen, V. Anklin, F. Wang, R. Feng, M. Gholami, K. Ling, L. Liu, J. Walter, H. Moghaddam, A. Kishore, J. Adamek, T. Mercado, J. Mallinson, S. Wandekar, S. Cagle, E. Ofek, G. Garrido, C. Lombriser, M. Mukha, B. Sun, H. R. Mohammad, J. Matak, Y. Qian, V. Peswani, P. Janus, Q. Yuan, L. Schelin, O. David, A. Garg, Y. He, O. Duzhyi, A. Älgmyr, T. Lottaz, Q. Li, V. Yadav, L. Xu, A. Chinien, R. Shivanna, A. Chuklin, J. Li, C. Spadine, T. Wolfe, K. Mohamed, S. Das, Z. Dai, K. He, D. von Dincklage, S. Upadhyay, A. Maurya, L. Chi, S. Krause, K. Salama, P. G. Rabinovitch, P. K. R. M, A. Sel-

van, M. Dektiarev, G. Ghiasi, E. Guven, H. Gupta, B. Liu, D. Sharma, I. H. Shtacher, S. Paul, O. Akerlund, F.-X. Aubet, T. Huang, C. Zhu, E. Zhu, E. Teixeira, M. Fritze, F. Bertolini, L.-E. Marinescu, M. Bölle, D. Paulus, K. Gupta, T. Latkar, M. Chang, J. Sanders, R. Wilson, X. Wu, Y.-X. Tan, L. N. Thiet, T. Doshi, S. Lall, S. Mishra, W. Chen, T. Luong, S. Benjamin, J. Lee, E. Andrejczuk, D. Rabiej, V. Ranjan, K. Styrce, P. Yin, J. Simon, M. R. Harriott, M. Bansal, A. Robsky, G. Bacon, D. Greene, D. Mirylenka, C. Zhou, O. Sarvana, A. Goyal, S. Andermatt, P. Siegler, B. Horn, A. Israel, F. Pongetti, C.-W. L. Chen, M. Selvatici, P. Silva, K. Wang, J. Tolins, K. Guu, R. Yogev, X. Cai, A. Agostini, M. Shah, H. Nguyen, N. . Donnaile, S. Pereira, L. Friso, A. Stambler, A. Kurzrok, C. Kuang, Y. Romanikhin, M. Geller, Z. Yan, K. Jang, C.-C. Lee, W. Fica, E. Malmi, Q. Tan, D. Banica, D. Balle, R. Pham, Y. Huang, D. Avram, H. Shi, J. Singh, C. Hidey, N. Ahuja, P. Saxena, D. Dooley, S. P. Potharaju, E. O'Neill, A. Gokulchandran, R. Foley, K. Zhao, M. Dusenberry, Y. Liu, P. Mehta, R. Kotikalapudi, C. Safranek-Shrader, A. Goodman, J. Kessinger, E. Globen, P. Kolhar, C. Gorgolewski, A. Ibrahim, Y. Song, A. Eichenbaum, T. Brovelli, S. Potluri, P. Lahoti, C. Baetu, A. Ghorbani, C. Chen, A. Crawford, S. Pal, M. Sridhar, P. Gurita, A. Mujika, I. Petrovski, P.-L. Cedoz, C. Li, S. Chen, N. D. Santo, S. Goyal, J. Punjabi, K. Kappaganthu, C. Kwak, P. LV, S. Velury, H. Choudhury, J. Hall, P. Shah, R. Figueira, M. Thomas, M. Lu, T. Zhou, C. Kumar, T. Jurdi, S. Chikkerur, Y. Ma, A. Yu, S. Kwak, V. Ähdel, S. Rajayogam, T. Choma, F. Liu, A. Barua, C. Ji, J. H. Park, V. Hellendoorn, A. Bailey, T. Bilal, H. Zhou, M. Khatir, C. Sutton, W. Rzadkowski, F. Macintosh, K. Shagin, P. Medina, C. Liang, J. Zhou, P. Shah, Y. Bi, A. Dankovics, S. Banga, S. Lehmann, M. Bredesen, Z. Lin, J. E. Hoffmann, J. Lai, R. Chung, K. Yang, N. Balani, A. Bražinskas, A. Sozanschi, M. Hayes, H. F. Alcalde, P. Makarov, W. Chen, A. Stella, L. Snijders, M. Mandl, A. Kärrman, P. Nowak, X. Wu, A. Dyck, K. Vaidyanathan, R. R. J. Mallet, M. Rudominer, E. Johnston, S. Mittal, A. Udathu, J. Christensen, V. Verma, Z. Irving, A. Santucci, G. Elsayed, E. Davoodi, M. Georgiev, I. Tenney, N. Hua, G. Cideron, E. Leurent, M. Alnahlawi, I. Georgescu, N. Wei, I. Zheng, D. Scandinaro, H. Jiang, J. Snoek, M. Sundararajan, X. Wang, Z. Ontiveros, I. Karo, J. Cole, V. Rajashekhar, L. Tume, E. Ben-David, R. Jain, J. Uesato, R. Datta, O. Bunyan, S. Wu, J. Zhang, P. Stanczyk, Y. Zhang, D. Steiner, S. Naskar, M. Azzam, M. Johnson, A. Paszke, C.-C. Chiu, J. S. Elias, A. Mohiuddin, F. Muhammad, J. Miao, A. Lee, N. Vieillard, J. Park, J. Zhang, J. Stanway, D. Garmon, A. Karmarkar, Z. Dong, J. Lee, A. Kumar, L. Zhou, J. Evens, W. Isaac, G. Irving, E. Loper, M. Fink, I. Arkatkar, N. Chen, I. Shafran, I. Petrychenko, Z. Chen, J. Jia, A. Levskaya, Z. Zhu, P. Grabowski, Y. Mao, A. Magni, K. Yao, J. Snaider, N. Casagrande, E. Palmer, P. Suganthan, A. Castaño, I. Giannoumis, W. Kim, M. Rybiński, A. Sreevatsa, J. Prendki, D. Soergel, A. Goedeckemeyer, W. Gierke, M. Jafari, M. Gaba, J. Wiesner, D. G. Wright, Y. Wei, H. Vashisht, Y. Kulizhskaya, J. Hoover, M. Le, L. Li, C. Iwuanyanwu, L. Liu, K. Ramirez, A. Khorlin, A. Cui, T. LIN,

M. Wu, R. Aguilar, K. Pallo, A. Chakladar, G. Perng, E. A. Abellan, M. Zhang, I. Dasgupta, N. Kushman, I. Penchev, A. Repina, X. Wu, T. van der Weide, P. Ponnappalli, C. Kaplan, J. Simsa, S. Li, O. Dousse, F. Yang, J. Piper, N. Ie, R. Pasumarthi, N. Lintz, A. Vijayakumar, D. Andor, P. Valenzuela, M. Lui, C. Paduraru, D. Peng, K. Lee, S. Zhang, S. Greene, D. D. Nguyen, P. Kurylowicz, C. Hardin, L. Dixon, L. Janzer, K. Choo, Z. Feng, B. Zhang, A. Singhal, D. Du, D. McKinnon, N. Antropova, T. Bolukbasi, O. Keller, D. Reid, D. Finchelstein, M. A. Raad, R. Crocker, P. Hawkins, R. Dadashi, C. Gaffney, K. Franko, A. Bulanova, R. Leblond, S. Chung, H. Askham, L. C. Cobo, K. Xu, F. Fischer, J. Xu, C. Sorokin, C. Alberti, C.-C. Lin, C. Evans, A. Dimitriev, H. Forbes, D. Banarse, Z. Tung, M. Omernick, C. Bishop, R. Sterneck, R. Jain, J. Xia, E. Amid, F. Piccinno, X. Wang, P. Banzal, D. J. Mankowitz, A. Polozov, V. Krakovna, S. Brown, M. Bateni, D. Duan, V. Firoiu, M. Thotakuri, T. Natan, M. Geist, S. tan Girgin, H. Li, J. Ye, O. Roval, R. Tojo, M. Kwong, J. Lee-Thorp, C. Yew, D. Sinopalnikov, S. Ramos, J. Mellor, A. Sharma, K. Wu, D. Miller, N. Sonnerat, D. Vnukov, R. Greig, J. Beattie, E. Caveness, L. Bai, J. Eisenschlos, A. Korchemniy, T. Tsai, M. Jasarevic, W. Kong, P. Dao, Z. Zheng, F. Liu, F. Yang, R. Zhu, T. H. Teh, J. Sanmiya, E. Gladchenko, N. Trdin, D. Toyama, E. Rosen, S. Tavakkol, L. Xue, C. Elkind, O. Woodman, J. Carpenter, G. Papamakarios, R. Kemp, S. Kafle, T. Grunina, R. Sinha, A. Talbert, D. Wu, D. Owusu-Afriyie, C. Du, C. Thornton, J. Pont-Tuset, P. Narayana, J. Li, S. Fatehi, J. Wieting, O. Ajmeri, B. Uria, Y. Ko, L. Knight, A. Héliou, N. Niu, S. Gu, C. Pang, Y. Li, N. Levine, A. Stolovich, R. Santamaria-Fernandez, S. Goenka, W. Yustalim, R. Strudel, A. Elqursh, C. Deck, H. Lee, Z. Li, K. Levin, R. Hoffmann, D. Holtmann-Rice, O. Bachem, S. Arora, C. Koh, S. H. Yeganeh, S. Pöder, M. Tariq, Y. Sun, L. Ionita, M. Seyedhosseini, P. Tafti, Z. Liu, A. Gulati, J. Liu, X. Ye, B. Chrzaszcz, L. Wang, N. Sethi, T. Li, B. Brown, S. Singh, W. Fan, A. Parisi, J. Stanton, V. Koverkathu, C. A. Choquette-Choo, Y. Li, T. Lu, A. Ittycheriah, P. Shroff, M. Varadarajan, S. Bahargam, R. Willoughby, D. Gaddy, G. Desjardins, M. Cornero, B. Robenek, B. Mittal, B. Albrecht, A. Shenoy, F. Moiseev, H. Jacobsson, A. Ghaffarkhah, M. Rivière, A. Walton, C. Crepy, A. Parrish, Z. Zhou, C. Farabet, C. Radebaugh, P. Srinivasan, C. van der Salm, A. Fidjeland, S. Scellato, E. Latorre-Chimoto, H. Klimczak-Plucińska, D. Bridson, D. de Cesare, T. Hudson, P. Mendolicchio, L. Walker, A. Morris, M. Mauger, A. Guseynov, A. Reid, S. Odoom, L. Loher, V. Cotruta, M. Yenugula, D. Grewe, A. Petrushkina, T. Duerig, A. Sanchez, S. Yadlowsky, A. Shen, A. Globerson, L. Webb, S. Dua, D. Li, S. Bhupatiraju, D. Hurt, H. Qureshi, A. Agarwal, T. Shani, M. Eyal, A. Khare, S. R. Belle, L. Wang, C. Tekur, M. S. Kale, J. Wei, R. Sang, B. Saeta, T. Liechty, Y. Sun, Y. Zhao, S. Lee, P. Nayak, D. Fritz, M. R. Vuyyuru, J. Aslanides, N. Vyas, M. Wicke, X. Ma, E. Eltyshev, N. Martin, H. Cate, J. Manyika, K. Amiri, Y. Kim, X. Xiong, K. Kang, F. Luisier, N. Tripuraneni, D. Madras, M. Guo, A. Waters, O. Wang, J. Ainslie, J. Baldridge, H. Zhang, G. Pruthi, J. Bauer, F. Yang, R. Mansour, J. Gelman, Y. Xu, G. Polovets, J. Liu, H. Cai,

- W. Chen, X. Sheng, E. Xue, S. Ozair, C. Angermueller, X. Li, A. Sinha, W. Wang, J. Wiesinger, E. Koukoumidis, Y. Tian, A. Iyer, M. Gurumurthy, M. Goldenson, P. Shah, M. Blake, H. Yu, A. Urbanowicz, J. Palomaki, C. Fernando, K. Durden, H. Mehta, N. Momchev, E. Rahimtoroghi, M. Georgaki, A. Raul, S. Ruder, M. Redshaw, J. Lee, D. Zhou, K. Jalan, D. Li, B. Hechtman, P. Schuh, M. Nasr, K. Milan, V. Mikulik, J. Franco, T. Green, N. Nguyen, J. Kelley, A. Mahendru, A. Hu, J. Howland, B. Vargas, J. Hui, K. Bansal, V. Rao, R. Ghiya, E. Wang, K. Ye, J. M. Sarr, M. M. Preston, M. Elish, S. Li, A. Kaku, J. Gupta, I. Pasupat, D.-C. Juan, M. Someswar, T. M., X. Chen, A. Amini, A. Fabrikant, E. Chu, X. Dong, A. Muthal, S. Buthpitiya, S. Jauhari, N. Hua, U. Khandelwal, A. Hitron, J. Ren, L. Rinaldi, S. Drath, A. Dabush, N.-J. Jiang, H. Godhia, U. Sachs, A. Chen, Y. Fan, H. Taitelbaum, H. Noga, Z. Dai, J. Wang, C. Liang, J. Hamer, C.-S. Ferng, C. Elkind, A. Atias, P. Lee, V. Listik, M. Carlen, J. van de Kerkhof, M. Pikus, K. Zaher, P. Müller, S. Zykova, R. Stefanec, V. Gatsko, C. Hirnschall, A. Sethi, X. F. Xu, C. Ahuja, B. Tsai, A. Stefanoiu, B. Feng, K. Dhandhanania, M. Katyal, A. Gupta, A. Parulekar, D. Pitta, J. Zhao, V. Bhatia, Y. Bhavnani, O. Alhadlaq, X. Li, P. Danenberg, D. Tu, A. Pine, V. Filippova, A. Ghosh, B. Limonchik, B. Urala, C. K. Lanka, D. Clive, Y. Sun, E. Li, H. Wu, K. Hongtongsak, I. Li, K. Thakkar, K. Omarov, K. Majmundar, M. Alverson, M. Kucharski, M. Patel, M. Jain, M. Zabelin, P. Pelagatti, R. Kohli, S. Kumar, J. Kim, S. Sankar, V. Shah, L. Ramachandruni, X. Zeng, B. Bariach, L. Weidinger, A. Subramanya, S. Hsiao, D. Hassabis, K. Kavukcuoglu, A. Sadovsky, Q. Le, T. Strohman, Y. Wu, S. Petrov, J. Dean, and O. Vinyals, “Gemini: A family of highly capable multimodal models,” 2024.
- [122] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, and T. B. Hashimoto, “Stanford alpaca: An instruction-following llama model.” https://github.com/tatsu-lab/stanford_alpaca, 2023.
- [123] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, “Robust speech recognition via large-scale weak supervision,” 2022.
- [124] S. Vemprala, R. Bonatti, A. Bucker, and A. Kapoor, “Chatgpt for robotics: Design principles and model abilities,” 2023.
- [125] M. Colledanchise and P. Ögren, “Behavior trees in robotics and ai,” July 2018.
- [126] M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. Smith, Y. Sun, and D. Weld, “Pddl - the planning domain definition language,” 08 1998.
- [127] Groot, “Groot behavior tree library,” 2024.

