A Framework for UAV Simulation Environment for Visual Navigation Deployment

Yijun Huang1
1. College of Software Engineering, Beihang University
Beijing, China
yjhuang@buaa.edu.cn

***Abstract*— Unmanned aerial vehicle (UAV) navigation is gaining more and more interest in both research and business. Real UAV experiments could be expensive and time-consuming. As an alternative, the validation and testing of UAV navigational algorithms using simulation is essential in the early phases of development when it is difficult to get high-precision and high-fidelity experiment data. In this proposal, we provide a framework for a visual navigational validation environment for unmanned aerial vehicles based on ROS2, PX4, and Gazebo. When implemented, the suggested simulator can provide non-expert end-users with a modular simulation environment that is faster and more reliable for verifying and advancing proposed UAV navigation solutions, such as visual SLAM (Simultaneous Localization and Mapping) algorithms.**

***Keywords*— Simulator, ROS2, PX4, Gazebo**

I. Introduction

Unmanned Aerial Vehicles (UAV) have many civil applications like search and rescue, delivery of goods, security and surveillance, and civil infrastructure inspection [1]. Navigation, including perception, localization, motion planning, and motion control, is the meta-capability for UAVs in successfully performing tasks. The Global Navigation Satellite System (GNSS) is the most common way of navigation. However, the signal of GNSS would be unavailable for indoor environments and vulnerable under forest canopies [2] and in urban or natural canyons.[3] Therefore, the application of UAVs in environments devoid of GNSS service has motivated research into GNSS-independent navigation solutions[4,5]. Nevertheless, although testing and verifying these solutions in realistic scenarios can be more faithful to the real world, it could also be expensive and even dangerous due to the complex hardware setup, safety precautions, and battery constraints [6]. Simulators, on the other hand, allow for the safe and economic development and testing of algorithms without the time-consuming and costly process of dealing with real-world hardware. An excellent simulation environment frequently possesses the following characteristics:

1. Conforms to physical principles, i.e., it can faithfully represent the dynamics in the real world;

2. Is fast in simulating, i.e., a large number of simulation data that can be used to verify the algorithm can be obtained quickly within limited resources and time;

3. Mirrors the real-life environment, i.e., maintains as much consistency with the actual scene as possible.

In practice, however, meeting all of the aforementioned attributes at the same time might be challenging, if not impossible. This is due to the fact that increasing scene features and physical characteristics increase the amount of calculation, lowering simulation speed, and implying that these aspects are clashing to some extent. [7]

Despite these constraints, current robotic communities have provided a plethora of very practical and useful simulation settings. Among these simulation environments, many for universal purposes emphasize attribute 1, which means focusing mostly on UAV dynamics with insufficient attention to environmental perception. Environment-integrated simulators, such as the Gazebo-based RotorS [8, 9], which is intended for tackling higher-level problems such as path planning and SLAM, focus more on environment perception so that visual navigational algorithms such as SLAM can be better validated. Furthermore, for better integration and deployment of perception or planning components, most academic developers have focused heavily on ROS[10]. However, because ROS is designed to use a master-centered node communication architecture, it is less robust and may result in the failure of the entire system if the master node crashes; additionally, ROS lacks adaptation for different operating systems, and ROS is not suitable for multi-UAV emulation due to memory management and network security limitations[11]. Furthermore, because the development and maintenance of the most recent ROS Noetic will end in 2025[12] and will be replaced by ROS2, it is worthwhile to design and implement a simulation platform based on ROS2, which can meet the requirement for long-term simulation iteration while making the simulation platform more stable and improving the performance and robustness of the entire system operation process[13].We will present a simulation environment based on ROS2, PX4[14], and Gazebo in this concept. Once created, end-users can easily convert the customized solution for visual navigation to real-world hardware. Software engineering ideas will also be implemented to improve quality and procedure management. We will present a simulation environment based on ROS2, PX4[14], and Gazebo in this concept. Once accomplished, end-users can easily convert the customized solution for visual navigation to real-world hardware. Software engineering ideas will also be implemented to improve quality and procedure management.

The remainder of this proposal is organized as follows: Section II presents a review of the literature to compare related works, and Section III introduces the proposed simulation framework with the design pattern and software procedure.

Section IV shows the estimated difficulties and possible solutions. Section V describes the metrics and measure for evaluating the performance of this simulator and give out expected results. Section VI drafts out the planned timeline for this scheme using the Gannt graph.

II. Research Aim

The purpose of this research is to, first, identify the simulation requirements for navigation solutions and specify the boundary as well as the scope of this system, second, propose an architecture for the simulation environment, and finally test the simulator and evaluate the accuracy and capacity for the deployment of visual navigation solutions.

III. Literature Review

The UAV simulation environment can roughly be divided into two categories: 1) universal robot simulation software and 2) UAV simulation environment based on the universal simulation platform for specialized objectives such as obstacle avoidance or network communication.

For universal simulators, Microsoft's Shah et al. developed AirSim[15] as a plugin for the Unreal Engine, which supports both hardware-in-the-loop (HITL) and software-in-the-loop (SITL) using flying controllers such as PX4, and Microsoft will later announce Project AirSim[16]. However, because it is a flight simulator rather than a robot simulator, it focuses on dynamics instead of perception. Likewise, since AirSim is tightly coupled to the UE rendering engine, it lacks scalability for simulation speeds and is difficult to integrate and migrate[17]. The jMavSim was first developed by Babushkin et al.[18] for testing PX4 firmware and devices can be easily applied with the PX4 firmware, but further expansion and application of this may be limited[19]. MIT's Guerra et al. [20] proposed the FlightGoggles, which are based on the powerful Unity implementation rendering engine, allowing researchers to accelerate software and algorithm development by avoiding evaluating complex and difficult-to-model interactions (such as aerodynamics, motor mechanics, and so on). Yunlong et al. developed the Flightmare[7] based on FlightGoggles, employing the Unity rendering engine and dynamic modeling separation method to achieve faster and more accurate dynamic simulation, but computational resource consumption remains an issue.

For a customized simulation environment, Ma et al. created a simulation environment[21] based on the AirSim, ROS, and PX4 SITL that combines simulation authenticity and portability. However, the simulator is incapable of carrying out high-level flight schedules. Based on Gazebo, ROS, and PX4, Xiao Kun et al. created the XTDrone simulation platform[22], which can effectively integrate different modules to achieve the relevant simulation requirements and adjust the simulation speed based on computer performance. Although XTDrone can perform some SLAM-related tasks, it is more focused on providing a general solution for UAVs than an off-the-shelf solution for visual navigation. Chen et al.[6] proposed an end-to-end simulator that combines localization, mapping, and planning kits with ROS-Gazebo-PX4 toolchain customization. However, the simulator only supports software in the loop (SITL) simulations, which means that well-tested algorithms has to be migrated to the drones with additional validations from the bottom up. Meanwhile, when the master node fails to function, ROS is less stable than ROS2, which has a decentralized architecture for its middleware. These platforms can meet the needs of their respective fields, and they all have the requisite characteristics for an ideal simulation environment, but they all have some of the following drawbacks: The configuration of the UAV indoor navigation algorithm verification is complicated; simulation accuracy is limited; migration is difficult; user customization and secondary development are difficult.

IV. Proposed Methodology

In this section, we design a simulator architecture based on ROS2, Gazebo, and PX4. Once implemented, the software will be able to better support user-customized visual navigation algorithm verification for UAVs and provide the corresponding user interface to evaluate the execution results of the corresponding algorithm, improving the efficiency and security of the algorithm verification, lowering the cost of the algorithm during the verification process, and shortening the development cycle of the UAV's algorithm. As a result, this simulation environment employs SITL first to eliminate software flaws in the early stages, followed by HITL to better mimic real-world execution.

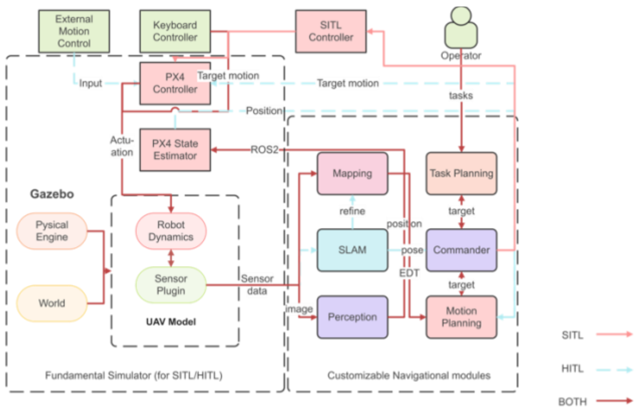


Figure 1. Draft architecture for the Simulator

As illustrated in Figure 1, we design an architecture for the simulation environment. The end-user parts are considered to show the interaction with the simulator in order to better illustrate the workflow of the proposed architecture. The simulator will support both SITL and HITL modes for smooth transitions from simulation to real-world UAV tasks. Depending on the algorithms used, the Gazebo provides world and UAV models, as well as custom-define sensor models. Users can then operate the UAVs with keyboard controllers to perceive the environment and compare the algorithm's estimation with the simulator's ground truth. The navigation modules are divided into separate modules that include localization, planning, and mapping components that the end user can customize for specific usage. Furthermore, because the simulator runs on ROS2, these modules communicate and coordinate with one another via ROS2 topics[24]. The communication between ROS2 and PX4[25] will also include PX4-supplied methods. In the early stages of SITL, the perception and planning modules, along with other components, run on the host. The user can begin by using QgroundControl to schedule flights and monitor drone status. The simulator will then instruct the sensors plugin in Gazebo to retrieve data from the environment and send it for navigation tasks such as perception and mapping. The situation in HITL would become more complicated with the introduction of environmental factors such as drone communications, weather, and human interruptions. In HITL mode, the navigation modules run on the UAV's onboard computer, and the Gazebo serves as an intermediary between the PX4 controller and the navigational modules.

The ability to abstract the system is required before using detailed software design patterns. High-level abstraction necessitates a thorough understanding of the entire system, including component interactions and the function of each component. As a result, the detailed architecture will be proposed and then refined later for a more seamless implementation.

V. DIFFICULTIES AND POSSIBLE SOLUTIONS

This section provides an overview of the potential problems and solutions. The first issue is that ROS and ROS2 are incompatible. To address this issue, we propose two possible solutions for running the two environments concurrently on a single host: 1) Use the ROS1bridge provided by the ROS developers; and 2) Use Docker containers to encapsulate the environments. The sophistication of the simulation environment itself would be the second barrier. To improve robustness and reusability, we intend to use the Model-Driven Engineering paradigm to decompose the workload for each module.

A. ROS and ROS2 Environment Incompatibility

The incompatibility of ROS and ROS2 is a barrier to implementing the simulator because the parallel nodes from ROS and ROS2 must run concurrently within a single simulator to make a trade-off between the abundance of ROS packages and the benefits of ROS2 such as higher robustness from distributed architecture. For example, the ROS package *rqt_bag* does not yet have a counterpart in ROS2[26]. To address these issues, two possible solutions are proposed: 1) Use the ROS1brigde to bridge the gap between ROS and ROS2; 2) Using Docker, create separate containers to hold two middleware.

1. ROS1brigde migration

The ROS community proposed ROS1bridge for the migration to ROS2 in 2015. The onboard computer's microRTPS agent acts as a bridge between PX4 and ROS2 messages and topics[27], allowing information (e.g., vehicle odometry) to be read from and commands (e.g., waypoints) to be written to the flight controller[28]. Supported by PX4, it allows ROS 2 nodes to interact with the PX4[29]. Despite its rapid iteration, the microRTPS is currently in use and could be adopted for further integration.

2. Docker encapsulation

Shunki et al.[30] devised a container-based method for migrating ROS-based mobile robots to ROS2 using separate Docker containers. Docker migration is more efficient than methods such as dual booting because it is a lightweight framework. The computer does not directly install ROS and/or ROS2, but rather implements the two ROS and/or ROS2 versions, as well as the corresponding Ubuntu version of the OS, as each container within Docker. Both environments are thus encapsulated in their respective containers, with communication capability reserved. The Docker method can be used further in this study to improve compatibility of different environments.

B. The complexity of the software development process

In large-scale software systems, the complexity of the software development process is unavoidable. A 2020 survey[31] revealed that 92 percent of robotic software developers were concerned about the robustness of current robotic software systems. As a result, the extensive use of object-oriented methods benefits software development engineering management and optimization. This method could aid in the development of a modular environment for large-scale software system implementation. Model-driven engineering is proposed as a promising solution to this problem. Model-driven engineering can be used in robotic software development procedures to improve robustness and reuse[32]. Instead of being driven by code, the model-driven process enriches the model during development until it is finally executable in the form of deployed software components[33]. The simulator will first be described in a model-based representation (platform independent model or PIM) that is independent of the underlying framework, middleware structures, operating systems, and programming languages in the proposed procedure [34]. Following validation of PIM's correctness, it will be transformed into a platform-specific model (PSM) that provides bindings for platform-specific refinements. The PSM will then be converted into a platform-specific implementation (PSI) [36].

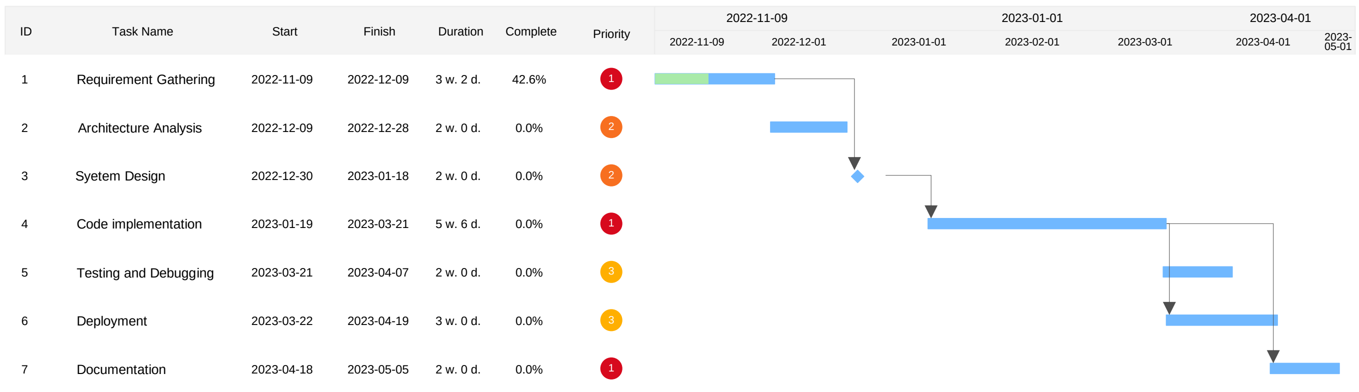
VI. EVALUATION AND EXPECTED RESULT

This section focuses on the simulator's evaluation metrics and expected test results. The test can be performed in two steps: the first will be a unit test, and the second will include an exhaustive test on well-known datasets. We will first define modular test units for the unit test in order to evaluate the functionality within each module and cover the stability of each one in the early stages. The comprehensive test is typically used to identify module faults.

Following the integration of each module, we will test the simulator's accuracy on the TUM[37], KITTY[38], and EuroC[39] datasets with SLAM-related algorithms such as ORB-SLAM2[40][41] and VINS-fusion[42] for monocular, stereo, and Lidar input to comprehensively compare the simulator's performance. Visual information is typically fused with IMU data in the UAV application using either a filter-based framework or an optimization-based framework. As a result, testing it with both visual and inertial data inputs would be more reliable. To quantify the results, we will use the trajectory comparison between our estimation and the ground truth[43, 44].

VII. Outline of the Working Plan

This section is about the simulator's strategic planning. A Gantt chart is used to illustrate the plan. The Gantt chart is a diagram that shows the relationship between various tasks[45]. We have the following tasks and subtasks for this scheme. During the requirement procedure, we will fully consider end-user requirements and create a prototype to specify and precisely define the system's interaction and expected behavior. The architecture of this simulator will then be further designed and refined, and a primary model for it will be presented. [46]. Figure 2 depicts the six stages of this research's work plan: 1)Requirement gathering, which includes a background check and competitor research. 2) The architecture analysis, which includes module decomposition and architecture refinement based on the specified requirements; 3) The system design, in which the UML will be used to build a coarse-to-fine structure of pyramidical models based on the architecture[47]. 4) Code implementation, which involves converting the defined models into executable codes. 5) Testing and debugging, when integration tests based on unit tests will be implemented to verify the consistency of each component before the comprehensive system test on datasets for validation; 6) The deployment process, in which the modules will be jigsawed and assembled into a simulator for use and further development; 7)The documentation process, in which all documents for requirements, design, deployment, and end-user instruction will be archived. This stage will also compile the previous stages into a paper and an open-source library.



ID	Task Name	Start	Finish	Duration	Complete	Priority
1	Requirement Gathering	2022-11-09	2022-12-09	3 w. 2 d.	42.6%	1
2	Architecture Analysis	2022-12-09	2022-12-28	2 w. 0 d.	0.0%	2
3	Syetem Design	2022-12-30	2023-01-18	2 w. 0 d.	0.0%	2
4	Code implementation	2023-01-19	2023-03-21	5 w. 6 d.	0.0%	1
5	Testing and Debugging	2023-03-21	2023-04-07	2 w. 0 d.	0.0%	3
6	Deployment	2023-03-22	2023-04-19	3 w. 0 d.	0.0%	3
7	Documentation	2023-04-18	2023-05-05	2 w. 0 d.	0.0%	1

Figure 2.The working plan for this research

References

[1] Shakhatreh, Hazim, Ahmad H. Sawalmeh, Ala Al-Fuqaha, Zuochao Dou, Eyad Almaita, Issa Khalil, Noor Shamsiah Othman, Abdallah Khreishah, and Mohsen Guizani. "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges." Ieee Access 7 (2019): 48572-48634.
[2] Cui, Jin Qiang, Shupeng Lai, Xiangxu Dong, Peidong Liu, Ben M. Chen, and Tong H. Lee. "Autonomous navigation of UAV in forest." In 2014 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 726-733. IEEE, 2014.
[3] Schmidt, G. T. "GPS based navigation systems in difficult environments." Gyroscopy and Navigation 10, no. 2 (2019): 41-53.
[4] Gyagenda, Nasser, Jasper V. Hatilima, Hubert Roth, and Vadim Zhmud. "A review of GNSS-independent UAV navigation techniques." Robotics and Autonomous Systems (2022): 104069.[3] Schmidt, G. T. "GPS based navigation systems in difficult environments." Gyroscopy and Navigation 10, no. 2 (2019): 41-53.
[5] Vanegas, Fernando, Kevin J. Gaston, Jonathan Roberts, and Felipe Gonzalez. "A framework for UAV navigation and exploration in GPS-denied environments." In 2019 ieee aerospace conference, pp. 1-6. IEEE, 2019.[3] Schmidt, G. T. "GPS based navigation systems in difficult environments." Gyroscopy and Navigation 10, no. 2 (2019): 41-53.
[6] Chen, Shengyang, Han Chen, Weifeng Zhou, C-Y. Wen, and Boyang Li. "End-to-end uav simulation for visual slam and navigation." arXiv preprint arXiv:2012.00298 (2020).[3] Schmidt, G. T. "GPS based navigation systems in difficult environments." Gyroscopy and Navigation 10, no. 2 (2019): 41-53.
[7] Song, Yunlong, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. "Flightmare: A flexible quadrotor simulator." In Conference on Robot Learning, pp. 1147-1157. PMLR, 2021.
[8] Furrer, Fadri, Michael Burri, Markus Achtelik, and Roland Siegwart. "Rotors—a modular gazebo mav

simulator framework." In Robot operating system (ROS), pp. 595-625. Springer, Cham, 2016.
[9] Koenig, Nathan, and Andrew 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, pp. 2149-2154. IEEE, 2004.
[10] Quigley, Morgan, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. "ROS: an open-source Robot Operating System." In ICRA workshop on open source software, vol. 3, no. 3.2, p. 5. 2009.
[11] Maruyama, Yuya, Shinpei Kato, and Takuya Azumi. "Exploring the performance of ROS2." In Proceedings of the 13th International Conference on Embedded Software, pp. 1-10. 2016.
[12] https://www.ros.org
[13] Macenski, Steven, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. "Robot Operating System 2: Design, architecture, and uses in the wild." Science Robotics 7, no. 66 (2022): eabm6074.
[14] Meier, Lorenz, 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), pp. 6235-6240. IEEE, 2015.
[15] Shah, Shital, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. "Airsim: High-fidelity visual and physical simulation for autonomous vehicles." In Field and service robotics, pp. 621-635. Springer, Cham, 2018.
[16] https://microsoft.github.io/AirSim/
[17] Hentati, Aicha Idriss, Lobna Krichen, Mohamed Fourati, and Lamia Chaari Fourati. "Simulation tools, environments and frameworks for UAV systems performance analysis." In 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), pp. 1495-1500. IEEE, 2018.
[18] A. Babushkin, jMavSim, Dec. 2020, [online] Available: https://github.com/PX4/jMAVSim.
[19] Collins, Jack, Shelvin Chand, Anthony Vanderkop, and David Howard. "A review of physics simulators for robotic applications." IEEE Access 9 (2021): 51416-51431.
[20] Guerra, Winter, Ezra Tal, Varun Murali, Gilhyun Ryou, and Sertac Karaman. "Flightgoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality." In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 6941-6948. IEEE, 2019.
[21] C. Ma, Y. Zhou and Z. Li, "A New Simulation Environment Based on Airsim, ROS, and PX4 for Quadcopter Aircrafts," 2020 6th International Conference on Control, Automation and Robotics (ICCAR), 2020, pp. 486-490, doi: 10.1109/ICCAR49639.2020.9108103.
[22] K. Xiao, S. Tan, G. Wang, X. An, X. Wang and X. Wang, "XTDrone: A Customizable Multi-rotor UAVs Simulation Platform," 2020 4th International Conference on Robotics and Automation Sciences (ICRAS), 2020, pp. 55-61, doi: 10.1109/ICRAS49812.2020.9134922.
[23] Xu, Guoliang, Zhiqun Yang, Wei Lu, and Luping Zhang. "Decentralized Multi-UAV Cooperative Search Based on ROS1 and ROS2." In International Conference on Autonomous Unmanned Systems, pp. 2427-2435. Springer, Singapore, 2021.
[24] Nyboe, Frederik Falk, Nicolaj Haarhøj Malle, and Emad Ebeid. "MPSoC4Drones: An Open Framework for ROS2, PX4, and FPGA Integration." In 2022 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1246-1255. IEEE, 2022.
[25] Jiang, Zhenhua, and Tejas Patil. "A Distributed Hardware-in-the-Loop Simulation Testbed for Swarms of Small Autonomous Vehicles." In AIAA AVIATION 2022 Forum, p. 4057. 2022.
[26] Xu, G., Yang, Z., Lu, W., Zhang, L. (2022). Decentralized Multi-UAV Cooperative Search Based on ROS1 and ROS2. In: Wu, M., Niu, Y., Gu, M., Cheng, J. (eds) Proceedings of 2021 International Conference on Autonomous Unmanned Systems (ICAUS 2021). ICAUS 2021. Lecture Notes in Electrical Engineering, vol 861. Springer, Singapore. https://doi.org/10.1007/978-981-16-9492-9_239
[27] Malle, Nicolaj H., Frederik F. Nyboe, and Emad Ebeid. "Onboard Powerline Perception System for UAVs using mmWave Radar and FPGA-Accelerated Vision." IEEE Access (2022).
[28] Bird, John J., Camron A. Hirst, Steven Valenzuela, Steve Borenstein, and Eric W. Frew. "An Autopilot Interface to Advance Fixed-Wing UAS Autonomy Research." In AIAA SCITECH 2022 Forum, p. 1852. 2022.
[29] Jiang, Zhenhua, and Tejas Patil. "A Distributed Hardware-in-the-Loop Simulation Testbed for Swarms of Small Autonomous Vehicles." In AIAA AVIATION 2022 Forum, p. 4057. 2022.
[30] Shibuya et al., "Seamless Rapid Prototyping with Docker Container for Mobile Robot Development," 2022 61st Annual Conference of the Society of Instrument and Control Engineers (SICE), 2022, pp.1063-1068, doi: 10.23919/SICE56594.2022.9905781.
[31] Sergio Garcı́a, Daniel Strüber, Davide Brugali, Thorsten Berger, and Patrizio Pelliccione.2020. Robotics software engineering: a perspective from the service robotics domain.In Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020).Association for Computing Machinery, New York, NY, USA, 593–604. https://doi.org/10.1145/3368089.3409743S.
[32] de Araújo Silva, Edson, Eduardo Valentin, Jose Reginaldo Hughes Carvalho, and Raimundo da Silva Barreto. "A survey of Model Driven Engineering in robotics." Journal of Computer Languages 62 (2021): 101021.
[33] Casalaro, Giuseppina Lucia, Giulio Cattivera, Federico Ciccozzi, Ivano Malavolta, Andreas Wortmann, and Patrizio

Pelliccione. "Model-driven engineering for mobile robotic systems: a systematic mapping study." Software and Systems Modeling 21, no. 1 (2022): 19-49.
[34] Benguria, Gorka, Xabier Larrucea, Brian Elvesæter, Tor Neple, Anthony Beardsmore, and Michael Friess. "A platform independent model for service oriented architectures." In Enterprise interoperability, pp. 23-32. Springer, London, 2007.
[35] Burt, Carol C., Barrett R. Bryant, Rajeev R. Raje, Andrew Olson, and Mikhail Auguston. "Quality of service issues related to transforming platform independent models to platform specific models." In Proceedings. Sixth International Enterprise Distributed Object Computing, pp. 212-223. IEEE, 2002.
[36] Müller, Marcus, Johannes Klöckner, Irina Gushchina, Alexander Pacholik, Wolfgang Fengler, and Arvid Amthor. "Performance evaluation of platform-specific implementations of numerically complex control designs for nano-positioning applications." International Journal of Embedded Systems 5, no. 1-2 (2013): 95-105.
[37] Tenorth, Moritz, Jan Bandouch, and Michael Beetz. "The TUM kitchen data set of everyday manipulation activities for motion tracking and action recognition." In 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, pp. 1089-1096. IEEE, 2009.
[38] Geiger, Andreas, Philip Lenz, and Raquel Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite." In 2012 IEEE conference on computer vision and pattern recognition, pp. 3354-3361. IEEE, 2012.
[39] Burri, Michael, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W. Achtelik, and Roland Siegwart. "The EuRoC micro aerial vehicle datasets." The International Journal of Robotics Research 35, no. 10 (2016): 1157-1163.
[40] Raúl Mur-Artal, J. M. M. Montiel and Juan D. Tardós. ORB-SLAM: A Versatile and Accurate Monocular SLAM System. IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147-1163, 2015.
[41] Raúl Mur-Artal and Juan D. Tardós. ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras. IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255-1262, 2017.
[42] Qin, Tong, and Shaojie Shen. "Online temporal calibration for monocular visual-inertial systems." In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3662-3669. IEEE, 2018.
[43] Kraft, Marek, Michał Nowicki, Adam Schmidt, Michał Fularz, and Piotr Skrzypczyński. "Toward evaluation of visual navigation algorithms on RGB-D data from the first- and second-generation Kinect." Machine Vision and Applications 28, no. 1 (2017): 61-74.
[44] Megalingam, R. Kannan, C. Ravi Teja, Sarath Sreekanth, and Akhil Raj. "ROS based autonomous indoor navigation simulation using SLAM algorithm." International Journal of Pure and Applied Mathematics 118, no. 7 (2018): 199-205.
[45] Tereso, Anabela, Pedro Ribeiro, Gabriela Fernandes, Isabel Loureiro, and Mafalda Ferreira. "Project management practices in private organizations." Project Management Journal 50, no. 1 (2019): 6-22.
[46] Lee, Sangkon, and Olga A. Shvetsova. "Optimization of the technology transfer process using Gantt charts and critical path analysis flow diagrams: Case study of the korean automobile industry." Processes 7, no. 12 (2019): 917.
[47] Ahmad, Tanwir, Junaid Iqbal, Adnan Ashraf, Dragos Truscan, and Ivan Porres. "Model-based testing using UML activity diagrams: A systematic mapping study." Computer Science Review 33 (2019): 98-112.