33RD CONGRESS
OF THE INTERNATIONAL COUNCIL
OF THE AERONAUTICAL SCIENCES
STOCKHOLM, SWEDEN, 4-9 SEPTEMBER, 2022

ICAS 2022 SWEDEN

INVESTIGATION OF SIMULATION FRAMEWORKS FOR THE EVALUATION OF AUTOMATED FLIGHT FUNCTIONS FOR EVTOL AIRCRAFT

Yves Lemmens[1], Casper Teirlink[1], Anoosh Hegde[1], David Pla Guerrero[1]

Gerardo Olivares[2], Harsh Shah[2]

[1] Siemens Digital Industries Software, Leuven, Belgium
[2] National Institute for Aviation Research, Wichita State University, Kansas

Abstract

This paper presents the capabilities of a Simcenter-based simulation framework to validate automated flight functions (AFF) for eVTOL aircraft systems. This is accomplished by coupling the modelled aircraft subsystems and sensors in Simcenter Amesim and Simcenter Prescan with external functionalities available in Robot Operating System (ROS). In particular, the Detect and Avoid (DAA) PX4-Avoidance algorithm and the simultaneous localization and mapping (SLAM) LeGO-LOAM algorithm are tested within the proposed framework. Simulations evaluate the performance of the implemented algorithms in urban scenarios.

Keywords: eVTOL; UAM; LiDAR; ROS; Siemens Simcenter.

1. Introduction

The introduction of autonomous urban air mobility vehicles is expected to revolutionize urban mobility. To ensure the safe operation of these autonomous vehicles, the flight management system needs to be extensively validated for diverse flight scenarios and operational conditions. Simulation models of the aircraft, sensors and environment are required to develop and validate the flight management system. Therefore, a digital twin is highly valuable as part of an efficient development process that ensures a safe product.

This study investigated and demonstrated a simulation framework based on Siemens Simcenter software that can be used to evaluate automated flight functions. Simcenter Amesim was used to model the flight dynamics together with propulsion systems and the navigation control loops. While Simcenter Prescan was used to create the simulation environment and model the LiDAR sensors used to detect features from the environment.

In this work, a Simcenter native solution made use of the C++ APIs of Simcenter Prescan and Simcenter Amesim to couple with ROS [1], enabling the usage of various open-source ROS packages.

2. C++ Coupling of Simcenter Prescan and Amesim

This first stage to create a flexible simulation using the Simcenter APIs involves creating the coupling between Simcenter Prescan and Amesim. In Figure 1, the framework architecture can be seen with the C++ executable performing the connection between both software. This shown setup can be used to evaluate the air vehicle system or sensor performance for scenarios with specified flight trajectories.

On the Simcenter Amesim side, a .dll import method was utilized. On compilation, the Amesim model generates a .dll library, which has its inputs and outputs predefined. This .dll file is then imported in C++ using the API functions available by Simcenter Amesim, which enables setting solver configuration parameters, calling simulation steps, assigning inputs and accessing outputs.

The Simcenter Prescan API allows setting up a master simulation model class. The simulation model class gives access to the simulation step and to the states of the simulated models. In this way, sensor information can be accessed, and the states of the air vehicle updated.

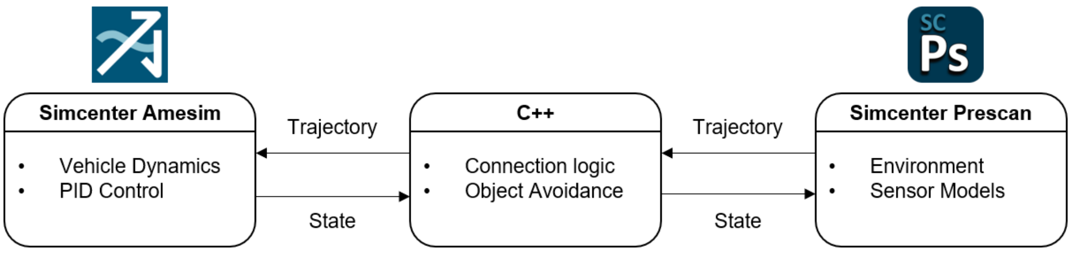

Figure 1 – C++ Simcenter framework architecture.

The Amesim .dll functions have been implemented inside the Simcenter Prescan master simulation model class, giving access to the drone's state computations from the Simcenter Prescan simulation step. In the simulation step function, the current waypoint coming from Simcenter Prescan is set as a Simcenter Amesim input and after that, a computation step is called. The new drone state is then read from the Amesim model, after which the Simcenter Prescan actor is updated.

This C++ framework has been extended to be compatible with external DAA and SLAM algorithms. To accomplish this, the mentioned API was integrated with ROS. Widely used in the field of robotics, ROS is an open-source software communication framework with a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robotic software across a wide variety of robotic platforms [2].

The implementation of ROS in the framework enhanced the simulation capabilities, introducing the possibility to validate avoidance and localization and mapping algorithms. The sensor models in Simcenter Prescan will provide the information of the environment required for the computation of these algorithms.

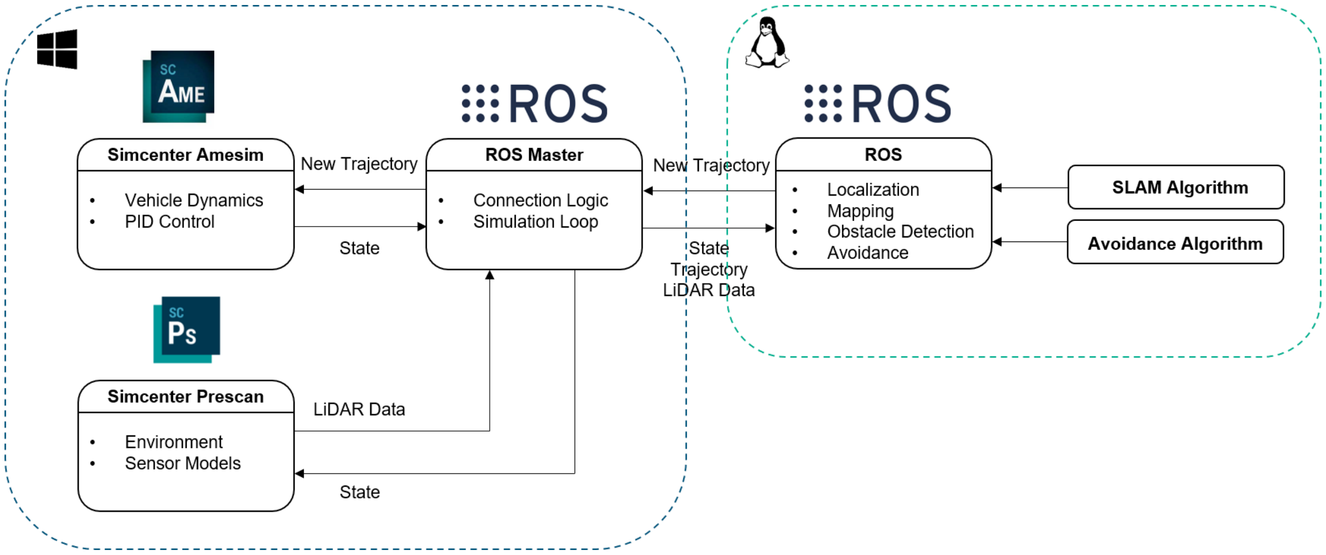

Figure 2 – Simulation framework architecture.

In Figure 2, a diagram of the Simcenter-ROS framework architecture can be seen. The simulation framework is executed in a Windows computer running Ubuntu in a virtual machine. In this manner, the full functionality of the ROS algorithms can be exploited within the Simcenter simulation. On the left part in Figure 2 resides the Windows operating system, with the Simcenter software and the ROS Master. In this side runs the ROS-core, a collection of nodes and programs required for a ROS-based system. On the right part, the Ubuntu operating system with the algorithms to be executed during the simulation is represented.

The ROS Master manages the data flow between all the processes. Once the simulation is started, it provides the configuration values to PX4 [3], which comprise the desired waypoints and parameter values for the DAA algorithm. It forwards to the algorithms the required state of the aircraft, being its position, velocity, acceleration and attitude. From Simcenter Prescan it receives the sensor output, which is also redirected to the Ubuntu side for the DAA and SLAM algorithms. Thereafter, it obtains the calculated safe waypoint from PX4 and the estimated location from the SLAM algorithm. The next safe waypoint to fly to is passed to Simcenter Amesim, which computes the state of the air vehicle and sends it to Simcenter Prescan.

3. Simulation Components

This chapter presents the different algorithms, LiDAR sensor and air vehicle models used in the presented simulation framework. First an overview of the aircraft model and the Simcenter Amesim block for the cosimulation will be shown. After that, a description of the LiDAR sensor characteristics will be detailed. Finally, both DAA and SLAM algorithms will be presented with its implementation architecture.

3.1 Aircraft Model

The framework was tested with an air taxi model based on the CityAirbus [4][5] as shown in Figure 3. The CityAirbus is an all-electric, four-seater, octocopter with 4 sets of 2 ducted counter-rotating propellers with each propeller coupled with an electric motor of 200kW. The system and flight dynamics model is shown in Figure 4.

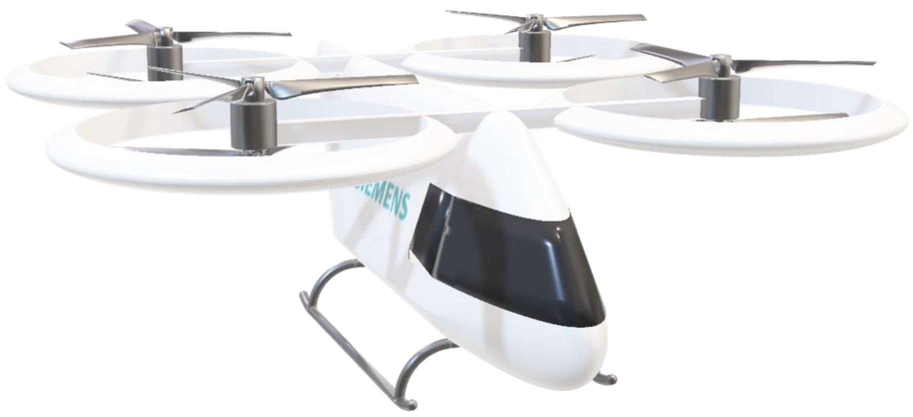

Figure 3 – CityAirbus air taxi model.

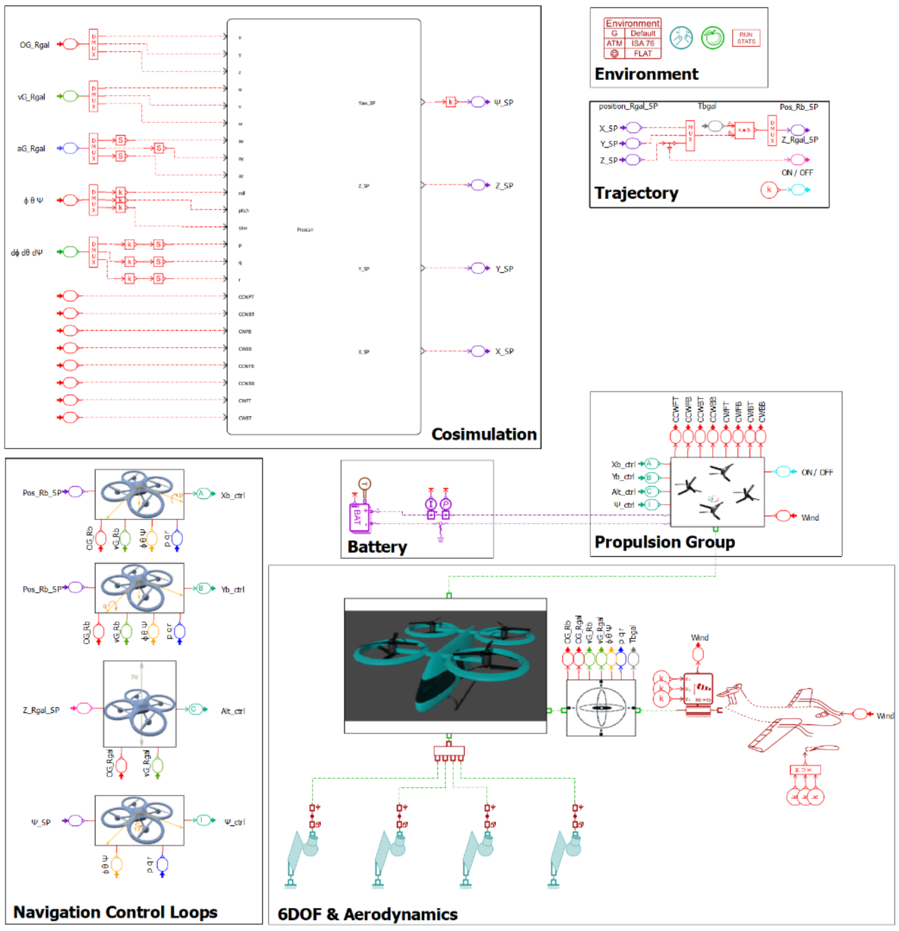

Figure 4 – Air taxi model in Simcenter Amesim.

The computed state of the aircraft will be passed to Simcenter Prescan and the ROS side to compute the obstacle avoidance and localization and mapping of the environment. On the left side of the cosimulation block, information of the position, velocity, acceleration, orientation, angular acceleration and thrust of each motor is given. Is important to notice that, the position shown at the top of the block will be used as ground truth to compare the pose estimation of the SLAM algorithm.

3.2 LiDAR Model

The LiDAR sensor from Simcenter Prescan models the operation of a real LiDAR sensor. Additionally, it offers the possibility to add configuration parameters to match the desired behaviour. In this case, featuring the VLP-16 LiDAR characteristics, it was specified a maximum and minimum vertical angle of 15 degrees and a sensor range of 100 meters. The scan pattern created 16 horizontal rings with a total number of 28928 points every 0.1 seconds.

During simulation, the selected sensor uses ray-tracing techniques to simulate the propagation of a beam through the simulated environment. Even though a physics-based LiDAR model was used, a sensor sub-model has been used that does not compute multi-bounce effects in order to increase the simulation speed, while maintaining an acceptable level of sensor fidelity.

3.3 PX4-Avoidance

The PX4-Avoidance algorithm [6] was selected as detection and avoidance algorithm for testing. The PX4-Avoidance is based on the 3DVFH+ algorithm [7], which is a 3D obstacle avoidance algorithm that uses an octomap internal data structure to represent the three-dimensional environment. The object avoidance module of the PX4-Autopilot is also available as a ROS node, allowing it to be implemented on a ROS framework. The PX4-Avoidance algorithm is part of the open source PX4-Autopilot [6] platform, a very popular autopilot compatible with multiple vehicle types including multi-copters, fixed-wing and VTOL aircraft.

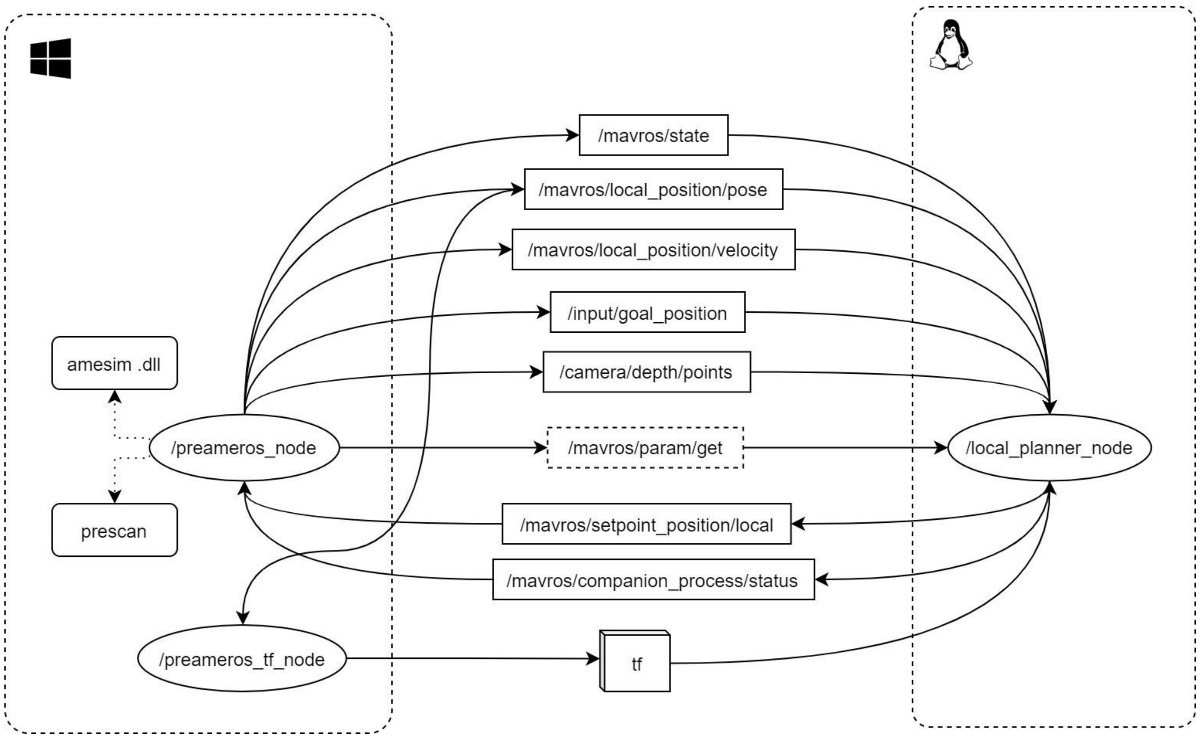

Figure 5 – PX4-Avoidance section of the ROS graph in the Simcenter-ROS framework.

In Figure 5, a more detailed ROS network graph of the Simcenter-ROS framework is shown including all the relevant ROS topics that are used between the master and the PX4-Avoidance node. A ROS framework can contain multiple nodes communicating data using the so-called publishers and subscribers. A ROS node, which is a process that performs computation, can send or receive data of a specific type to or from a ROS topic. There are three main nodes within the framework. The so called preameros_node, enables the communication between the Simcenter software and its coupling with ROS. The node in charge of computing the localization and mapping in the simulation is the lego_loam_bor and finally, the local_planner_node, executes one of the three object avoidance types that are supported by the PX4-Avoidance package.

The preameros_node employs both the Simcenter Amesim and Simcenter Prescan APIs and publishes to a number of ROS topics the expected data by the local_planner_node. These topics are mavros/local_position/pose, mavros/local_position/velocity, camera/depth/points and input/goal_position corresponding to drone position and attitude, drone velocity, LiDAR output and desired waypoint respectively.

Note that several topics are named with "/mavros". This relates to the fact that the supported simulation setup of PX4-Avoidance ROS node is using the PX4-Autopilot firmware in the loop. This would mean that the autopilot firmware handles the control loops and path following of the waypoints and eventually outputs the thrust control signals. The PX4-Autopilot firmware however is not a ROS node or package but operates using a different communication framework called MAVLink. MAVLink is a lightweight messaging protocol for communication with drones, both with ground control stations and between drone hardware components. It also follows a publisher subscriber design pattern similar to ROS. MAVROS is a ROS package that acts as a wrapper for MAVLink communication, essentially allowing MAVLink communication over a ROS network, which explains the expected ROS topic naming.

In the case of this study, it has been chosen to only use the PX4-Avoidance module and not the PX4-Autopilot firmware, whose tasks are being accomplished by the PID control loops build into the air taxi Simcenter Amesim model. In this manner, a major function of the preameros_node node is to emulate the presence of the PX4-Autopilot. That includes sending connection state messages, which happens over the mavros/state topic, as well as providing the PX4 configuration parameters that normally come from the firmware. This is accomplished by including a ROS parameter server on the topic mavros/param/get which on callback from the PX4-Avoidance node provides the settings of the autopilot. Examples of these parameters are the desired cruise speed or the maximum horizontal acceleration of the air vehicle.

The preameros_tf_node provides a ROS reference frame transform broadcaster on the ROS network. The PX4-Avoidance node expects to receive reference frame transformations from a ROS tf instance. The ROS tf package is used to keep track of multiple coordinate frames organized in a tree shape. It provides transformation matrices between any two defined reference frames.

The reference frames that are defined inside the tf broadcaster are the LiDAR sensor position, the inertial frame and the body frame. The first two frames are static transformations as the LiDAR sensor positions can be statically defined inside the body frame of the vehicle. The body frame itself has to be updated every iteration which is why the preameros_tf_node is also subscribed to the mavros/local_position/pose topic.

3.4 LeGO-LOAM SLAM

As an extension to the object avoidance, an optimized version of the LeGO-LOAM [8] algorithm has been employed in this study as a localization and mapping algorithm. It combines LiDAR sensor information together with odometry to perform mapping and pose estimation, capable of to achieve real-time estimations on low-powered embedded systems.

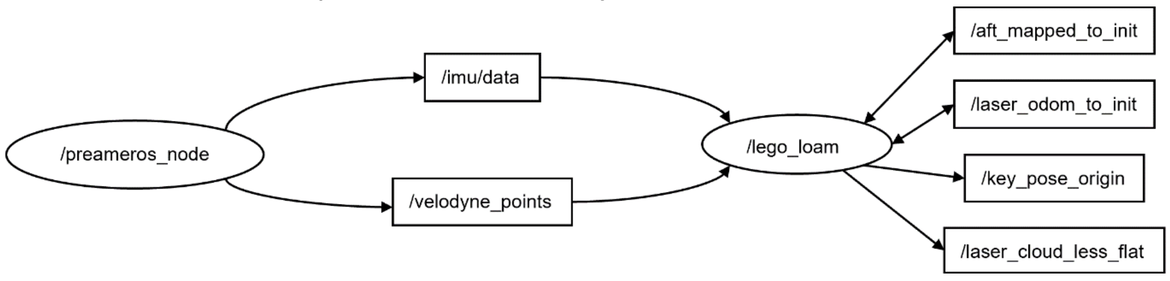

Figure 6 – SLAM section of the ROS graph in the Simcenter-ROS framework.

LeGO-LOAM is developed based on the presence of a ground plane in its segmentation and optimization steps. First, a point cloud segmentation is applied to filter out noise, then a feature extraction obtains distinctive planar and edge features. A two-step Levenberg-Marquardt optimization method uses the planar and edge features to solve different components of the six degree-of-freedom transformation across consecutive sensor scans [9].

More precisely, the algorithm takes point cloud data from a Velodyne VLP-16 LiDAR in a horizontal position and IMU (Inertial Measurement Unit) data as inputs. This is provided following the same procedure as previously explained for the avoidance algorithm. In this case, the output of the LiDAR sensor will be passed through the /velodyne_points topic and the simulated IMU through the /imu/data topic as shown in Figure 6. The pose estimation of the air vehicle is given through the /key_pose_origin, used to plot the estimated trajectory along the computed map of the environment. The remaining topics are required for the mapping computation and visualization.

4. Simulation Results

The results of the simulations focused mainly on demonstrating the abilities of the framework, not as an extensive validation of the containing models or algorithms.

4.1 PX4 Avoidance Algorithm

Multiple scenarios were created, progressing from simple to more complex situations for the object avoidance algorithm. The reference flight trajectory of the vehicle will be specified by a series of waypoints. The flight sequence starts with a vertical take-off until the altitude of the first waypoint has been achieved. After that, it will transition to forward flight until reaching the goal waypoint, marked in the figures as a yellow dot. When the final position has been reached, the aircraft will finally descend vertically.

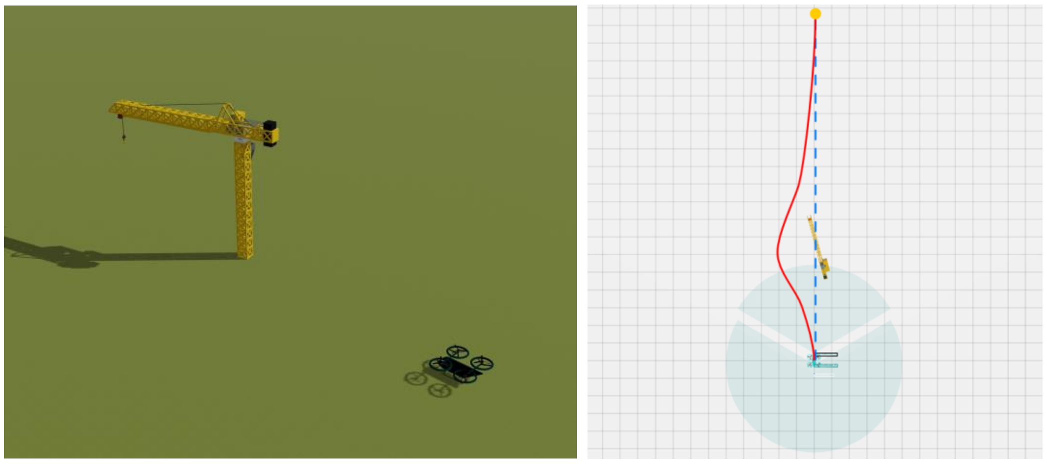
Figure 7 – First simple Simcenter Prescan scenario and trajectory result plot.

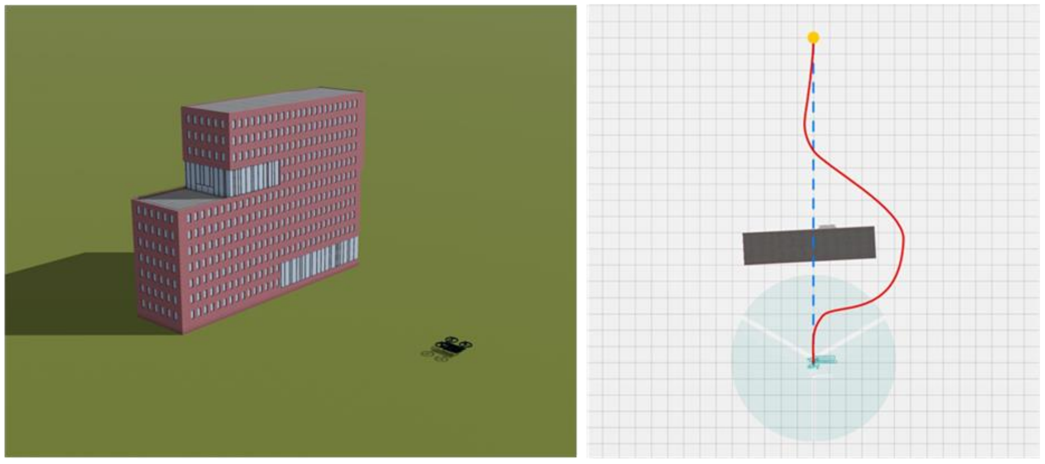
Figure 8 – Second simple Simcenter Prescan scenario and trajectory result plot.

In Figure 7 and Figure 8 are shown the results for the first test scenarios. On the left, the visualization in Simcenter Prescan and on the right, the plots show the predefined trajectory as a blue dashed line, while the red line shows the path the vehicle followed. From both simple situations, it can be seen how the air vehicle was able to detect the obstacles along its trajectory, compute and travel an avoidance maneuver to finally reach the goal position.

The next scenario shown in Figure 9 presented additional obstacles, what implies a higher demand for the computation of the avoidance trajectory. The avoidance path fulfilled the evasion of the obstacles and reached the final commanded waypoint. Nevertheless, in order to compute such trajectory, the velocity needed to be limited to 5 m/s. Otherwise, the algorithm did not calculate an avoidance trajectory on time to ensure the safety of the vehicle.

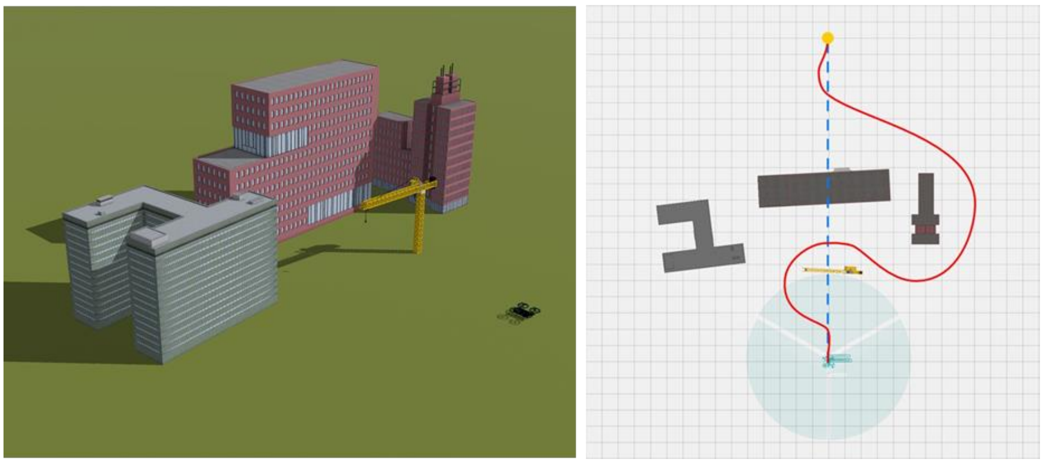
Figure 9 – Complex Simcenter Prescan scenario and trajectory result plot.

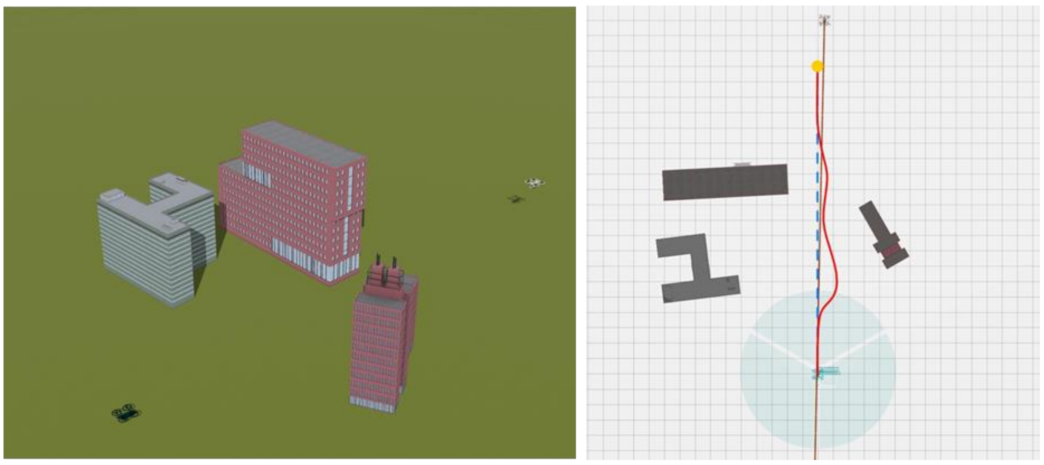
Figure 10 – Moving obstacle Simcenter Prescan scenario and trajectory result plot.

The final scenario shown in Figure 10 contains a moving obstacle, in this case a similar air taxi drone, flying in a straight line. The maximum horizontal speed was again set to 5m/s to ensure the avoidance path calculation. As it can be seen, the collision with the moving drone was avoided by an aggressive right turn.

4.2 SLAM Algorithm

The SLAM algorithm, regardless of not being specifically developed for aerial vehicles, performed well in the simulations. Nevertheless, if the flight altitude was set too high, where the ground could not be detected by the LiDAR sensor, the localization did not perform as expected.

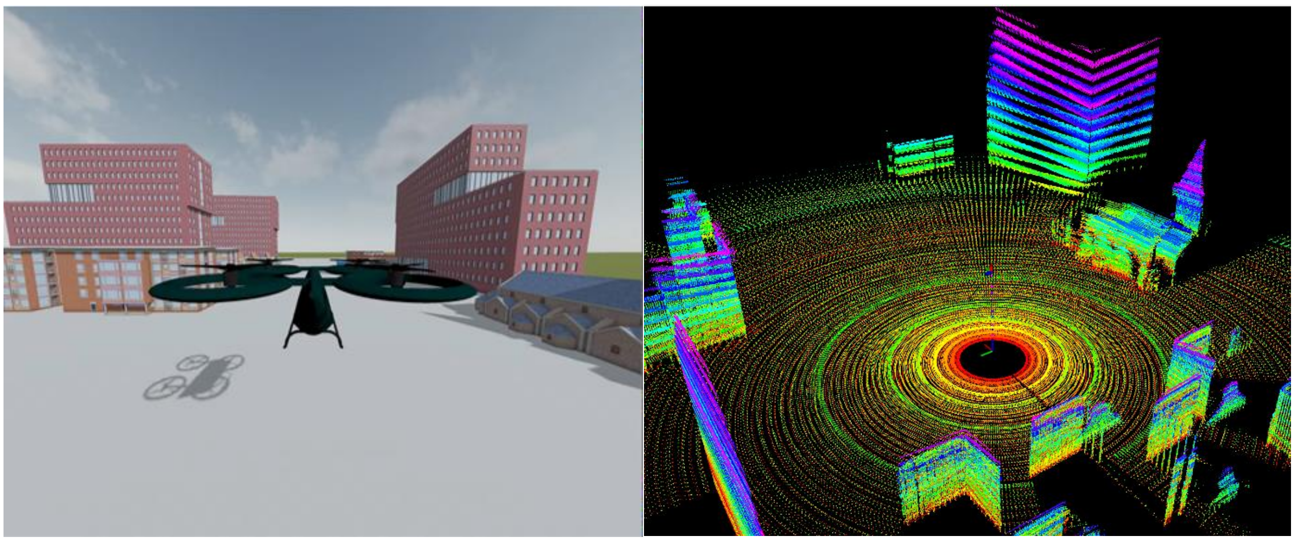
Figure 11 – Simcenter Prescan urban scenario and SLAM point cloud map plot.

In Figure 11 it is shown on the left the urban scenario used to test the localization and mapping algorithm. In this case, the commanded path to the air vehicle encountered no obstacles. In this manner, the computation of avoidance trajectory will not be triggered, which would influence the performance of the SLAM algorithm.

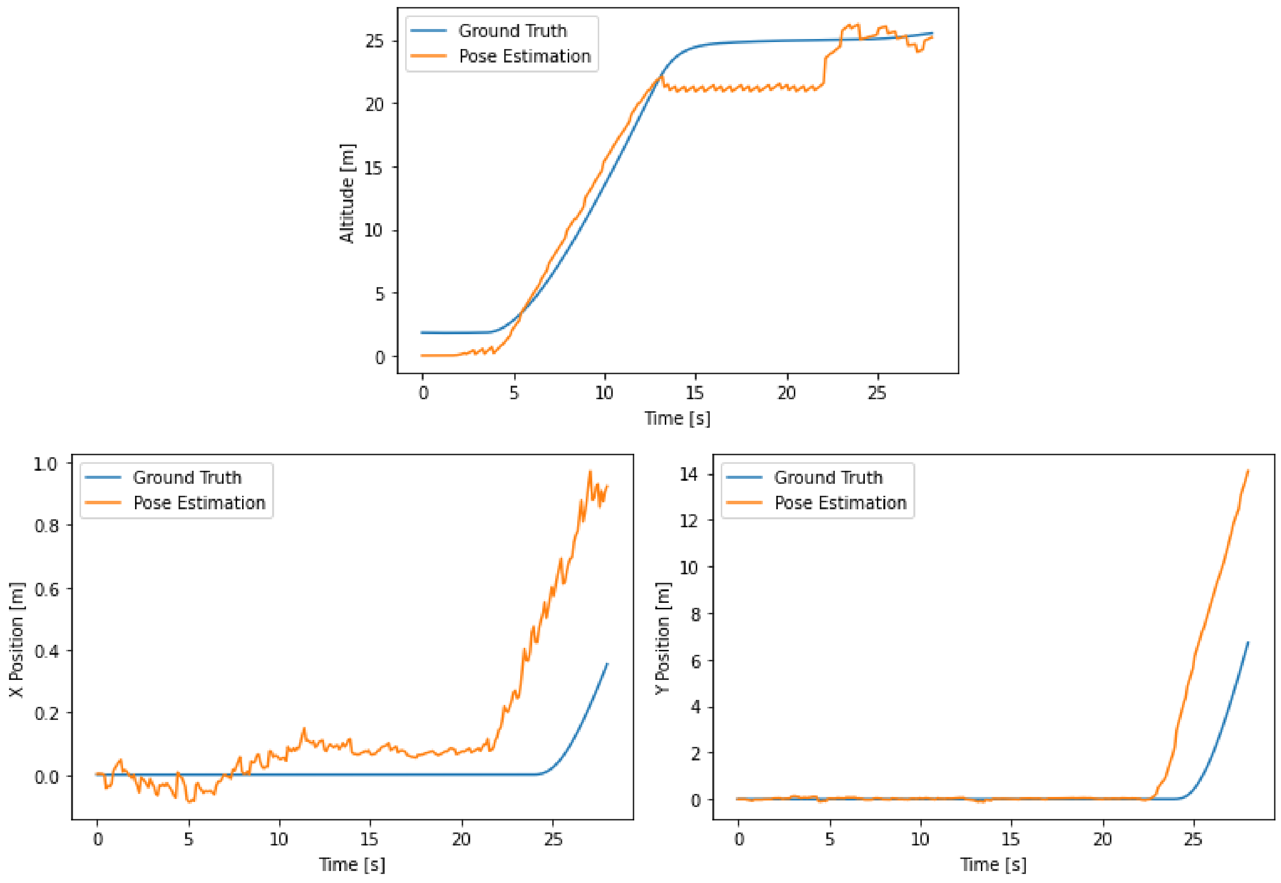

Figure 12 – SLAM pose estimation and ground truth plots.

In Figure 11 the output map of the computed point map is displayed. The different dot colors obtained from the of the detection of the environment features represent the ring height from the scan pattern of the LiDAR sensor. In pink squares it is represented the pose estimated by the SLAM algorithm.

The commanded trajectory required first a gain of altitude and secondly, a straight flight between the buildings. From Figure 12 it can be seen how the LeGO-LOAM estimated correctly the trajectory followed by the air vehicle in the simulation.

5. Conclusion

The objectives of this study were to create a Simcenter native simulation framework and enable its coupling with external functionalities available in ROS. The test scenarios evaluated the open-source PX4-Avoidance DAA algorithm and the LeGO-LOAM SLAM algorithm for an eVTOL aircraft. This demonstrated that the framework can be used to efficiently evaluate automated flight functions for urban air mobility vehicles that are available in ROS or developed as a C++ implementation.

6. Future Work

The Simcenter-ROS framework has been developed with the use of automatic design space exploration tools in mind. This enables the use of a design exploration and optimization tools that interfaces with engineering and CAD software. In the future, the Simcenter-ROS framework will be coupled to Simcenter HEEDS to automatically identify and investigate critical scenarios for autonomous flight missions using ROS DAA or controller modules.

The Simcenter-ROS framework has been implemented using the ROS1 framework. The main reason for using ROS1 has been the compatibility of the PX4-Avoidance package which at the time of writing did not have a version compatible with ROS2. ROS1 however is not meant to be used for real-time simulation performance while ROS2 was developed with real-time performance in mind. Since both Simcenter Amesim and Simcenter Prescan support real-time simulations, porting the framework to ROS2 would result in a real-time simulation framework that could be used in Hardware-in-the-loop testing e.g DAA and SLAM algorithms on embedded hardware. These setups can be beneficial in the validation process of controllers and autopilots.

7. Contact & Acknowledgment

The authors can be contacted at yves.lemmens@siemens.com.

The research presented in this paper has been supported by the MARLOC research project and has received funding from the Flemish government through the VLAIO ICON Programme (ref. HBC.2020.2600)

8. Copyright Statement

References

[1] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. In Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics, May 2009.

[2] ROS. About ros. Retrieved from: https://www.ros.org/about-ros/ Accessed: May 2022.

[3] PX4. Px4 autopilot, Retrieved from: https://px4.io/. Accessed: October 2021.

[4] Airbus CityAirbus. Retrieved from: https://en.wikipedia.org/wiki/Airbus_CityAirbus. Accessed: October 2021.

[5] Anton F. “Hybrid-electric propulsion systems for aircraft”, Tag der Deutschen Luft- und Raumfahrtregionen, Potsdam, 10 September 2019. Retrieved from: https://www.bbaa.de/fileadmin/ user_upload/02-preis/02-02-preistraeger/newsletter-2019/02-2019-09/02_Siemens_Anton.pdf. Accessed: October 2021.

[6] PX4 Obstacle Detection and Avoidance. Retrieved from: https://github.com/PX4/PX4-Avoidance#local-planner. Accessed: May 2022.

[7] Vanneste, Simon & Bellekens, Ben & Weyn, Maarten. 3DVFH+: Real-Time Three-Dimensional Obstacle Avoidance Using an Octomap. CEUR Workshop Proceedings, 2014.

[8] Davide Faconti. LeGO-LOAM-BOR, 2018. Retrieved from: https://github.com/facontidavide/LeGO-LOAM-BOR/tree/speed_optimization.

[9] T. Shan and B. Englot. LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 4758-4765, doi: 10.1109/IROS.2018.8594299.