A Multi-UAV Co-Simulation Environment for Safety and Performance Analysis

Leydson Silva, Ewerton Salvador, Alisson Brito
Universidade Federal da Paraiba (UFPB)
Centro de Informatica
Joao Pessoa, Paraiba, Brazil
Email: alisson@ci.ufpb.br, ewerton@ci.ufpb.br

Jose Sousa Barros
Instituto Federal da Paraiba (IFPB)
Campus Guarabira
Guarabira, Paraiba, Brazil

Vivek Nigam
fortiss
Guerickestraße 25, 80805
Munich, Germany
Email: vivek@fortiss.org

***Abstract*—Co-Simulation can provide important insights on how well mission strategies perform. Simulations can provide insights on which strategies could provide safety guarantees because of the high investments necessary for the creation of a multi-UAV system. We propose a co-simulation framework of multiple UAVs for the analysis of missions in order to evaluate different flight strategies with great efficiency. To make the environment as realistic as possible, we introduced external threats such as winds, sectors with flight restrictions, and aircraft crashes. We validated our framework using a case study of an air surveillance system, where two types of flight strategies comparing two different flight strategies. We can analyze trade-offs between the flight strategies, in terms, of chances of UAV collision, running out of energy, and performing tasks.**

***Index Terms*—Multi-UAV Simulator, Co-Simulation, Flight Strategy Valuation.**

I. Introduction

Unmanned Aerial Vehicles (UAVs) are aircraft that operate without the need for a pilot or flight crew, being able to fly autonomously or remotely controlled via radio. Some UAV applications that one can highlight include the defense [1], commercial activities[2] [3] [4], agriculture [5] [6], scientific approach [7], governmental activities [8], and personal enjoyment [9]. Despite this importance, there is little tool support for the design of flight strategies and allow the realistic analysis and evaluation of safety level.

Some efforts have been undertaken to propose adequate simulation tools of UAV systems [10], [11], [12]. In these works, we used Ptolemy to model the flight strategies and the simulator SITL/Ardupilot [13] to represent the behavior of the UAV. Nonetheless, these cited works only approach the design and evaluation of flight strategies that use a single UAV.

The critical challenging of developing co-simulation frameworks for multi-UAV scenarios is the high degree of complexity involved. While for single UAV scenarios, co-simulation involves a single UAV and model with the decision, the use of multi-UAVs scenarios requires the synchronization of multiple instances of UAVs. Moreover, the co-simulation master needs to gather and consolidate the data in the different UAV simulation instances in order to determine, e.g., whether UAVs have collided.

This work proposes a multiple-UAV, co-simulation environment to analyze the performance of the flight strategies. To this end, the work [11] has been modified to support the simulation of multiple UAVs. Thus we propose changes in the structural model, the logical model, and the conceptual model of the work in question. Our environment uses the simulators and work strategies mentioned above, but we have added new features such as navigation visualization, collision avoidance, and restricted flight zones. Each UAV in the scenario is simulated using the SITL/Ardupilot environment. Our framework then uses the High-Level Architecture (HLA) [14] to gather and exchange the telemetry data between the different instances of SITL/Ardupilot.

In Section II, we review related work, to then in Section III to discuss the overall architecture of our framework. In Section IV, we present our experimental results, dicussing them in Section V. We conclude the paper in Section VI by pointing to future work.

II. Related Work

Corner and Lamont [15] propose a high-performance computing parallel discrete event simulation system. This system was implemented using a layered approach, where the SPEEDES parallel environment acts as the lowest layer, and the top layer consists of a swarm behavior model [16]. Although the proposal is significant to the area of Multi-UAV simulators, the authors' implementation could not achieve the desired speedup expected from the parallel approach.

Wei, Madey, and Blake presented an agent-based control framework [17], which employs multiple UAV agents for local task scheduling. They developed a simulation as a proof-of-concept for the proposed framework. The simulation tool was useful for validating the framework, but it is not likely to be easily extended to simulate approaches other than the one proposed by the authors.

Zema et al. introduce an architecture for a simulator that is capable of implementing distributed networked control systems, called CUSCUS (CommUnicationS-Control distribUted Simulator) [18]. The proposed architecture takes advantage of two already validated solutions: the FL-AIR (Free AIR Framework) simulator and NS-3. The integration between the simulators works via a shared memory data structure. One disadvantage of this type of integration is the fact that this

is a non-standardized way of connecting different simulators, which can make it harder to extend the proposed architecture.

D-MUNS is a distributed network simulator for multiple UAVs [19]. The system uses two simulators: RotorS and NS-3. A novel real-time synchronizing algorithm is proposed to solve the synchronizing problem between the distinct simulators. This work differs from our proposal in the sense that we use HLA for synchronization between different simulators.

Mason et al. proposed a formal framework for analyzing adaptive autonomous aerial vehicles [20]. The framework integrates three main components: Executable Formal Specification of UAV Behavior, UAV Simulator, and Statistical Model Checker. Despite some similarities with our work, this proposal lacks a collision detection system.

A high fidelity simulator for a quadrotor UAV using ROS and Gazebo is proposed by Zhang *et al.* [21]. The proposed simulator implements dynamic models and system navigation besides other features. On the other hand, this proposal does not include support for multi-UAV systems.

Our proposed solution is capable of integrating different well-validated simulators in order to compose a single parallel simulation that can provide results with a high level of realism.

TABLE I
TABELA DE COMPARAÇÃO COM O ESTADO DA ARTE

Works	**Mult. UAV**	**Co-sim**	**Simul.**	**Uncert.**	**Coll.**
Corner and Lamont [15]	✓	✗	SPEEDS	✗	✗
Wei, Madey, and Blake [17]	✓	✗	Own tool in Java	✗	✗
Zema et al. [18]	✓	✗	NS-3 and FL-AIR	✗	✗
La, Park, and Kim [19]	✓	✗	RotorS and NS-3	✗	✗
Mason et al. [20]	✓	✓	Maude and SITL	✓	✗
Our work	✓	✓	Ptolemy and SITL	✓	✓

III. ARCHITECTURE

The proposed environment uses two simulation tools, both with different functions. The first one is the Ptolemy II [22]. Ptolemy II is an open-source heterogeneous modeling tool that provides a virtual laboratory for experimenting with design technologies for cyber-physical systems. We use Ptolemy II for modeling the environment and for developing strategies to be used in the multi-UAV systems. Ptolemy is responsible for executing the defined strategy and indicating the path that UAV should follow through commands. The collision detection module is also run by Ptolemy.

The other tool used was Software in the Loop (SITL/Ardupilot) [13]. SITL/Ardupilot is a free code simulator, developed from a compilation of codes of an autopilot written in C++. The SITL uses MAVLink, a protocol for data exchange between the UAV and its components. SITL is responsible for the visual UAV representation, and MAVLink sends telemetry data from UAVs to the environment. The SITL is responsible for UAV physics, including components and sensors. The SITL executes the movements of the UAVs, according to Ptolemy commands.

In order to provide a communication mechanism capable of joining these tools in a co-simulation environment, we use the High-Level Architecture (HLA). According to Gomes *et al.* [23], Co-Simulation can be defined as a set of simulators that form a single system capable of performing a global simulation. The HLA is a standard of the Institute of Electrical and Electronics Engineers (IEEE). Three documents define it: the first deals with the general framework and main rules [24], the second concerns the specification of the interface between the simulator and the HLA [14], and the third is the model for data specification (Object Model Template) transferred between the simulators [25]. The HLA treats as federates the subsystems of the environment composed of the different tools. The set of federates that make up the whole environment is called federation. HLA also provides a communication and synchronization infrastructure among the federates called Run-Time Infrastructure (RTI).

For our purpose, HLA is a communication bridge for data exchange among heterogeneous systems. The SITL/Ardupilot sent the telemetry data from the UAVs to the HLA, which in turn passed on to Ptolemy II. Then the data was processed and used for decision making that would depend on the employed strategy. The result of the processing would be the next command that the UAV would follow, transmitted to the

SITL through the HLA. Figure 1 describes the architecture of the environment. The left Federate represents Ptolemy II and implements the communication interface with the JCerti RTI. JCerti is an interface that implements HLA for the Java programming language, which we use in Ptolemy. On the right is the representation of the SITL federation, implementing PyHLA. The RTI is responsible for synchronizing the messages and transmitting data between the federates that make up the federation.

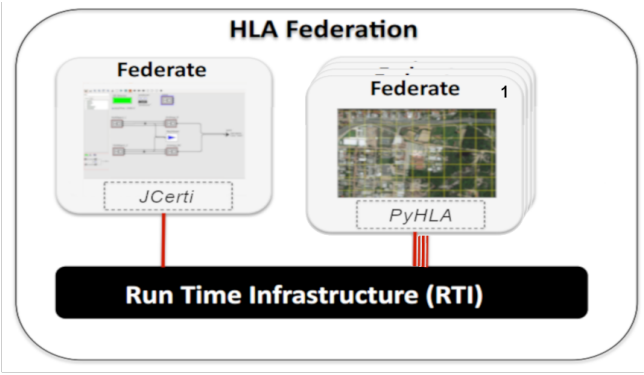

Fig. 1. Architecture of the Simulation Environment

A. Drone Simulator

With the ability to simulate various types of vehicles and still provide telemetry data, the SITL/Ardupilot works as the drone simulator. We chose quadcopters due to their ease of taking off and landing, and for their maneuverability. Simplistically, we can define the simulator in three distinct phases: initial, median, and final. For each phase, a function: take off, fly, and land.

In the initial phase, the SITL/Ardupilot verifies all the components doing the security check. Then he sends a signal that he is ready to take off. The drone proceeds with the takeoff go up to the predetermined altitude and proceed to the middle phase. In the next phase, the drone moves forward in order to complete its mission. In this case, the drone is given Ptolemy II commands, via HLA, to define the next move. The movement commands that the Ptolemy II sends to the SITL are: mvE, where it would move to the east, mvW, movement to the west, mvN, and mvS, respectively, moves north and south. The complete command that the Ptolemy II sends to the HLA to pass to the SITL/Ardupilot was UAVId + mvX, where the UAVId is the identifier of the drone that would receive such command, and the mvX specifies the movement that the drone would take. Lastly, in the final phase, the drone would receive the command to land. There are four moments that the landing function is called. The first moment is when the drone completes its mission and returns to base. The second moment is when the UAV reaches the limit of 80% of battery consumption. In this case, the UAV abandons the mission and returns to base. The third moment occurs when the drone reaches the threshold of 95% of battery consumption, which is the critical energy level. If the UAV is far from the base and cannot return until it reaches this limit, it immediately lands wherever it is to preserve the equipment. The last moment is when there is a collision. In this case, the command is used to take the drone out of the simulation.

We used the SITL/Ardupilot in the visual representation of the UAVs. However, the tool is limited to simulate only one vehicle per graphical interface. We solved this problem by creating different instances for each drone. Figure 1 (right side) also presents an SITL/Ardupilot instance, in graphical mode, simulating a quadcopter. Even though adding multiple UAVs to the environment makes it difficult to see the number of open instances. So we added a map component, which shows at simulation time all simultaneously running UAVs and their paths.

Figure 2 shows the sequence diagram of the first iteration of the proposed environment. We can observe seven main elements. The actor who represents the user who uses the environment. The Ptolemy II, HLA, and SITL tools. The producer and soft-constraints controllers, which are the HLA JCerti and PyHLA APIs, used by Ptolemy II and SITL, respectively.

The wind occurrence algorithm is executed in the final step before the UAV receives the move command. It is a pseudo-random algorithm that selects the presence of wind and its intensity (weak, moderate or strong). The wind is simulated as a force that pushes the UAV from east to west.

Moreover, the Map object, used by Ptolemy II and visualized throughout the simulation. User interaction with the proposed environment is minimal. It starts the environment execution, fills the map with targets, if it is not already completed, and completes the planning. From this step, the simulation begins. At the end of the simulation, the environment automatically saves the information obtained from the entire simulation and individually for each UAV.

A collision detection system has been added to the environment. The system works this way: a circular area of 2.6m (which can be configured) of diameters around the center of the UAV has been delimited. This area is the impact limit, i.e., when crossing this area, there is no way to escape from the collision. Then each UAV calculates its distance to all others present in the simulation and checks for the intersection of areas. Similarly, each UAV checks the distance from each other and all flight restriction zones in the environment and checks whether it has crossed that zone itself. Ptolemy retains UAV location information in global variables, accessed by all UAVs. Unlike collisions between UAVs, collisions with Restricted Areas do not take into account the impact limit area, only taking the location of the UAV in the calculation. If a collision is detected, the colliding UAV is removed from the simulation.

B. Strategy Configuration

Ptolemy II is a modeling tool that uses actor-oriented models. Actors are like classes of components assigned to perform a given function. In this way, we have created a model for an environment with the ability to simulate multiple drones. The main feature of this mode is reusability. Then, we created

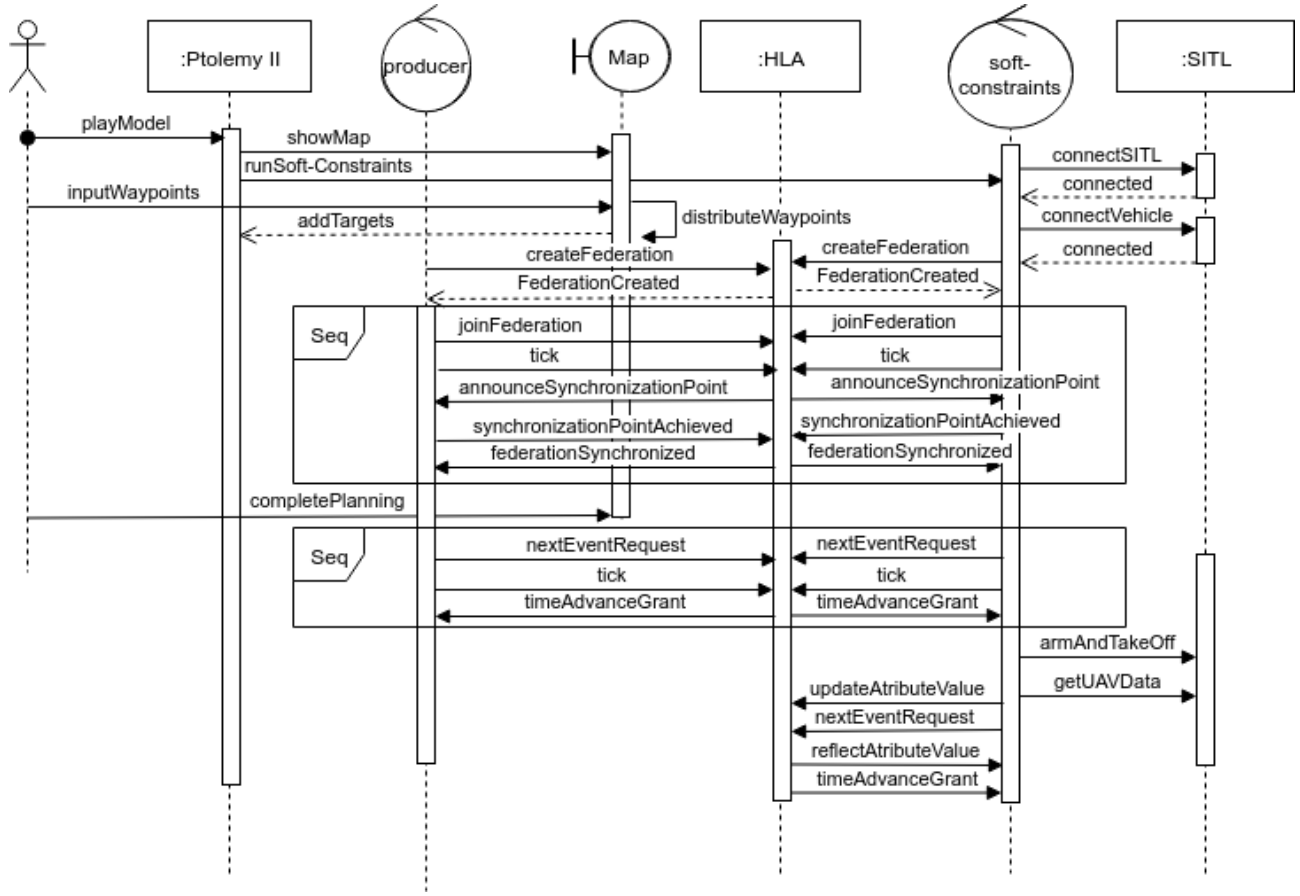

Fig. 2. Sequence diagram of initial environment run

two distinct strategies and defined which strategy to use at a given time.

Regarding the components, we adapt an actor called robot to represent the drone. It received three parameters: Id, which was used to identify which drone would be sending data; Battery, essential to record the charge consumption; and GPS, which informed the current location of the UAV. For each UAV added to the simulation, it would be necessary to add an instance of the robot actor to the model.

Besides, two composite actors were adapted, which are actors with implemented sub-models, referring to the strategies that the drones would use in their missions. The first actor, called Strategy A, receives the three parameters of the instance of the robot class, and calculate battery consumption and location. Then Strategy A forwards the UAV to the next target, as chosen on the map, following from the first to the last point and returning to the base. The actor named Strategy B, besides the mentioned actors, has some more actors that calculate the distances that the UAV will go to reach the targets and sets a score for each distance. This score serves as the basis for choosing the next target. An actor was created representing the map with missions progress during execution in different routes run by the drones. Figure 1 (left side) shows an example of configuration of the model responsible for the simulation environment.

The HLAObject actors are instances of the robot class and must relate to the actor of the strategy, represented by the actor strategy B, in addition to connecting the GPS port to the map actor, represented by MapViewer. The Producer actor, along with the director, is responsible for implementing the HLA for Ptolemy. The result of the strategies processing is a command passed to the goto component, which sends via HLA to SITL/Ardupilot.

As an example of how a Strategy Actor works internally, we can visualize its structure in Figure 3. The actor receives three parameters: id, battery, and position. The Energy Actor records the battery consumption of each UAV separately and sends it to the Combine Actor. The RankPoints Actor calculates the distance from all the visit points assigned to the UAV, and it assembles a score. This score, together with the ordered points positions, are passed to the RankActions Actor, which defines the action that the UAV should take. Actor Combine also receives this data, and it is responsible for combining this data and, prioritizing the limit of battery consumption, passes to the Selector Actor. It is the Selector Actor that chooses which action the UAV should take, and then it sends the result to the HLA to relay to the SITL/Ardupilot. The Log Actor records all the actions. There are still two actors that show on

the screen at runtime the individual consumption of the UAVs and the actions that each drone performs.

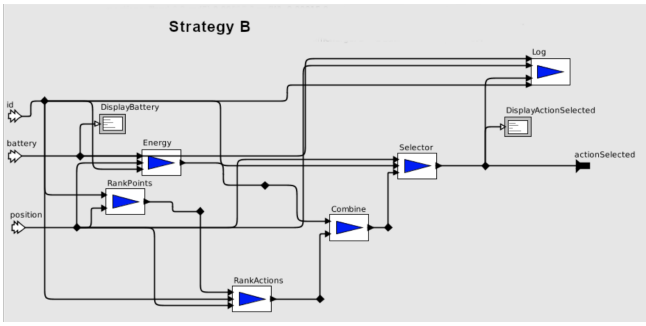

Fig. 3. Strategy B modeled using Ptolemy

C. Environment Configuration

Before running the simulation, the environment must be configured. In order to do so, it is necessary to edit the init.properties file located in the root folder of the project. There are several parameters to be edited, such as the base location from which the UAVs starts the simulation, where latitude and longitude coordinates must be used. There are also parameters, like the altitude on takeoff and the number of UAVs available in the simulation. The latter should have the same number of instances contained in the model. When running the environment for the first time, it is necessary to define which waypoints define the navigation of the drones. The first waypoint marked on the map will always be where the drones will land after the mission is completed. The other points mark the targets to be visited by the drones. The marked waypoints are recorded in a file called targets.txt, which will be remembered in subsequent runs. For the definition of new points, the target.txt file must be deleted before starting the simulation. One can also define areas with flight restrictions. For this, it is necessary to select the restricted area option, in the text box on top of the map. Next, with the right mouse button, one must select on the map, the place where the flight restriction should be imposed. Restricted areas are registered in the file restrictedAreas.txt, and in case changes to the areas are needed the file must be recreated. At the end of the simulation, the outputs are stored in files, as distance traveled, battery consumption, the number of captured photos, together with the location from which the photo was taken, and the taken path.

IV. Experiments

In order to validate the proposed tool, let us consider the case of an airborne surveillance plane containing visiting points distributed randomly. These points are divided equally for each drone. We have defined a total number of 24 visitation points. We configured two scenarios: the first with an area free of obstacles, and the second with restricted flight areas, as shown in Figure 4. These areas were defined as impact areas, where a collision would be detected should any drones cross this area. We also added support for wind occurrence probabilities to increase the realism of the simulation.

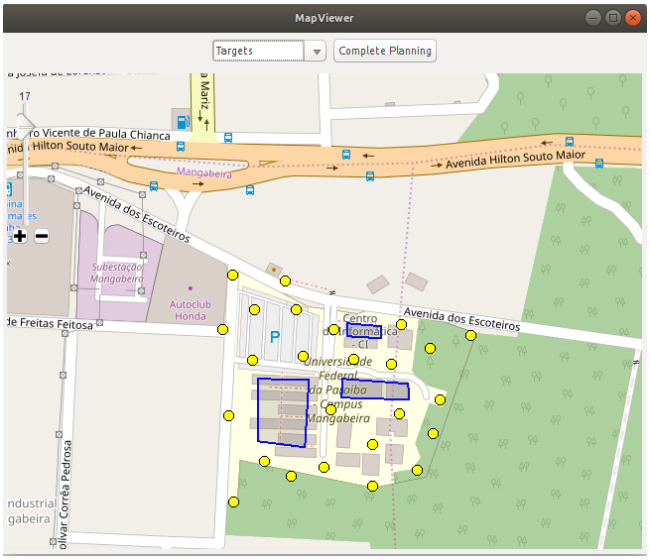

Fig. 4. Map of Waypoints in Restricted Area

Figure 4 shows how the visit points were arranged. The circles in yellow indicate the points of visitation. The polygons in blue are the restricted areas.

We use the two strategies used by [11] and adapt to guide the drone missions. In the first strategy, called Strategy A, the drones followed the points of visitation sequentially, according to the way they were defined on the map. In the second strategy, called Strategy B, the UAVs would first visit the points closest to its location.

The mission of the UAVs would be to visit all designated target points on the map and capture an image of the location, then return to the base. If a UAV reaches 80% of battery consumption, it should leave the mission and return to base. Moreover, if a UAV that cannot return to base and reaches 95% of battery consumption, it would immediately land, to safeguard the equipment.

The data of all the simulations were recorded in files and later transformed into a table. For each scenario, all strategies were simulated separately, and some drones were used, ranging from one to four. Besides, for each possibility of wind occurrence, three simulations were performed, which resulted in a total of 336 simulations.

V. Results and Discussions

As results, the simulator generates the metrics, remaining battery of drones, the total distance of the mission, and the number of collisions that occurred during simulations. Moreover, it was defined as the concepts of success and failure in the simulations to measure how safe they are. Success occurs when a UAV traverses all of the assigned visitation points and returns to base with battery charge higher than 5%. Failure occurs in all other situations. A critical battery level (5% or less) is an indication of a forced landing, which is not

considered a success. Figure 5 shows the number of successes obtained by the amount of employed UAVs.

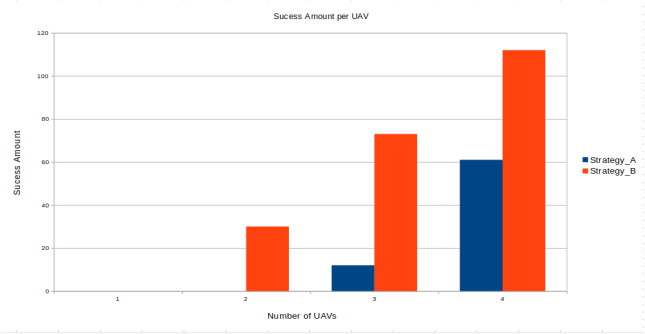

Fig. 5. Success rate per UAV

The blue bars indicate the amount of success obtained by the UAVs using Strategy A. The red ones indicate the amount of success in Strategy B. The maximum number of successes that could be obtained for the strategy simulations with 1, 2, 3 and 4 UAVs, respectively, were 42, 84, 126, 168. It can be noted that using only 1 UAV was unsuccessful and with 2 UAVs only success by strategy B. Although none of the strategies succeeded in achieving 100% utilization, it can be noted that strategy B was superior to strategy A.

Another metric that was studied in this work is the battery consumption of the UAVs. Missions where the UAV spent more time in flight, or the ones that contained points of visitation more distant from each other, led to higher battery consumption. In Figure 5, UAVs that used Strategy B had more success in completing their missions than those employing Strategy A. This turns the Strategy B safer than Strategy A. In fact, the total remaining battery of the drones that used Strategy B was 41.83%, against 41.07% of Strategy A. The average amount of remaining battery, considering the scenarios of free and restricted areas, is presented in Figure 6. It can be observed that Strategy A, characterized by the blue color, was superior to Strategy B. This is due to the number of collisions at the beginning of the simulation.

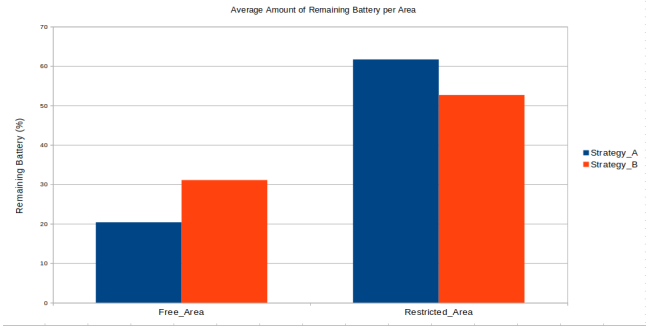

Fig. 6. Average remaining battery per area

Concerning the remaining battery charge concerning the percentage of wind occurrence, it can be observed in Figure 7 that the consumption increases with higher probabilities of wind because the drones correct their trajectories as the winds push them. The graph shows the average amount of remaining battery concerning the percentage of probability of wind occurrence, observing the strategies and the areas. The colors show the percentage of occurrence of wind, with the darker blue meaning no wind, continuing until dark green, which means 15% of probability of wind. Low battery consumption in restricted areas is justified by the high rate of collisions in these areas, influenced to some extent by the wind.

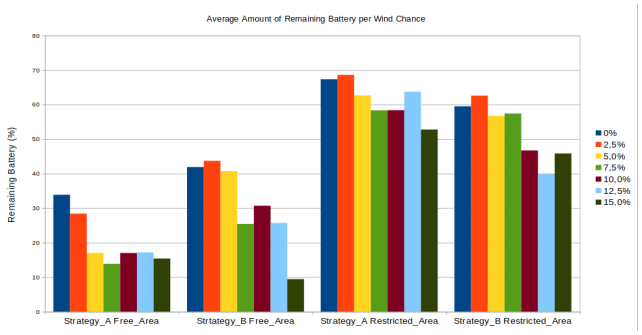

Fig. 7. Average remaining battery with wind influence

Another studied metric was the total distance traveled, where the most efficient strategy is the one with the lowest total distance among its missions. Again, Strategy B was superior in this metric, with a total distance traveled was about 240km. Strategy A was about 270km, due to the dynamism of the choice of its points of visitation. More specifically, even though Strategy A had closer points of visitation, it was guided by a deliberate choice, regardless of distance, while in Strategy B the choice of the next point was dynamically calculated considering the distance factor.

When the zoning of the scenarios is taken into account, it is observed that, in restricted areas, the distances traveled are smaller. This can be explained by the number of collisions, which ended up decreasing the total distance traveled, since the drones did not finish their missions. The distance traveled by Strategy A was superior to that of Strategy B. This provides evidence that there were more collisions at the beginning of the simulations in Strategy A than in Strategy B.

Moreover, in the case of total distance traveled, one can verify the influence of the wind in the simulations. Figure 8 shows the total distance traveled concerning the probability of occurrence of wind, regarding the two studied strategies. In darker blue, the percentage of wind occurrence is 0%, while in dark green it is 15% (highest percentage). One notices that, with some exceptions, the higher the occurrence of wind, the greater its influence on the total distance of the flight course.

One of the most important metrics is the total amount of collisions. In this study, we did not employ a collision avoidance method. For scenarios with restricted areas, the drones try to fly around, but it changes the route only when a restricted area is detected from approximately 2 meters away. For this evaluation, it was defined that a collision occurs regardless of the type of impact point, a UAV, or a restricted

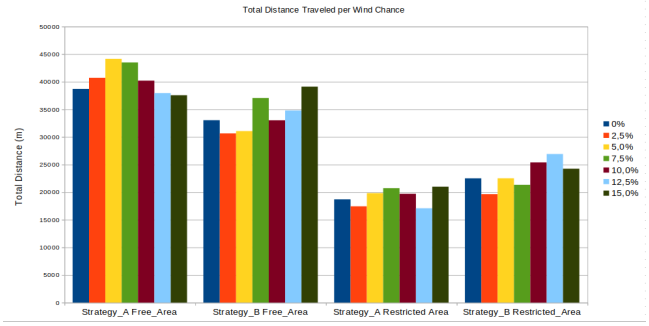

Fig. 8. Graphic Total Distance Traveled per Wind Chance

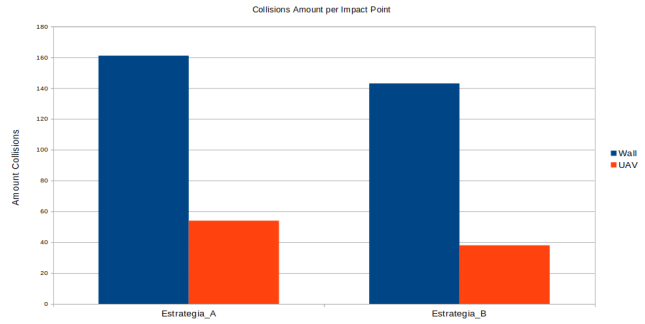

Fig. 9. Collisions amount per impact point

area. For this metric, Strategy A had 215 total collisions, while in Strategy B, there were 181.

It was also analyzed the collision by different impact points. Figure 9 shows the number of collisions for the point of impact. It is observed that, in the two strategies, the number of collisions occurred in the wall is the triple of those between UAVs. These collisions occur caused by the incidence of the wind acting against the UAV, and by the late detection of the impact area, resulting in the drone not being able to maneuver in time.

Concerning the battery consumption of the drones, one can find out in Figure 10 that the average of the remaining battery charge per quantity of UAVs tends to gradually increase with the introduction of a more considerable amount of drones in the simulations. However, a notable exception can be identified in the usage of Strategy A, when only one UAV was used. The explanation resides in a spatial question; that is, the last point of visitation that the drone flew over before the remaining battery reached 20% was near the landing base station.

Finally, the usage of Strategy B was the one that achieved better safety criteria according to the set of data obtained by the proposed simulation environment. The proposed environment fulfills its role of simulating and analyzing several different strategies, using missions with multiple UAVs across different scenarios, and generating precise information that can be useful in real decision-making or testing.

Finally, the use of strategy B achieved the best performance with this data set obtained by the proposed simulation environment (see Table II). The performance of each strategy was

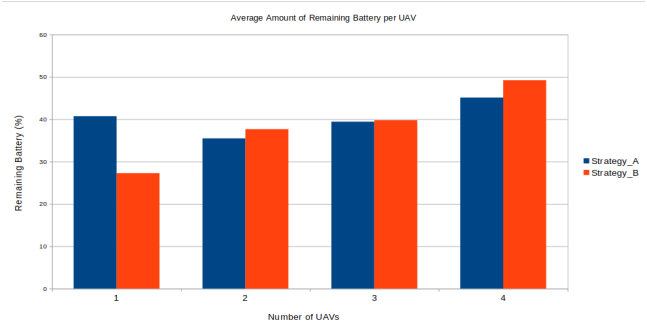

Fig. 10. Average remaining battery per UAV amount

measured using the sum total of each metric, ie, based on the remaining battery per area, and the result was used as the parameter to measure the final performance.

It can be concluded that the proposed environment fulfills its role of simulating and analyzing several different strategies, using missions with multiple UAVs between different scenarios and generating conclusive information that can be useful in decision making or actual testing.

VI. Conclusion

In this work, we propose a simulation tool for the evaluation of missions with multiple UAVs using co-simulation. It is based on the work [11] and adapted to support multiple UAVs, adding collision management and visualization of multiple vehicles. In order to validate the environment, we defined a scenario where the drones, using different strategies, should fulfill their missions. To get as close as possible to the real world, we added some, such as restricted areas, collisions, and the influence of the wind on the drones.

The results obtained from this work allow the evaluation and discussion of better methods of mission planning and strategies to be adopted. As demonstrated, the proposed environment was able to simulate multiple drones by different scenarios, collecting data for later analysis, and helping to evaluate the best strategy to be employed.

As future work, we intend to create a communication channel between the UAVs, so that they can exchange messages between them at certain distances, informing their future positions and points of visitation. By doing so, the drones will be able to create even more dynamic strategies. Furthermore, it is intended to create a control mechanism to mitigate collisions, using the communication channel itself to achieve this goal.

Acknowledgment

This work is supported by The Brazilian National Council for Scientific and Technological Development (CNPq) and the Higher Education Improvement Coordination (CAPES).

TABLE II
SUCCESS OF EACH STRATEGY

Metric	Strategy A	Strategy B
Success per number of UAVs	✗	✓
Number of photos	✗	✓
Number of photos with wind	✗	✓
Remaining energy	✗	✓
Remaining energy with wind	✗	✓
Travelled distance	✗	✓
Travelled distance with wind	✗	✓
Number of collisions	✗	✓

REFERENCES

[1] H. Mao, H. Feng, F. Zhang, and X. Zhao, "Reconnaissance and strike integrated uav's path planning in autonomous attack," in *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*. IEEE, 2016, pp. 282–288.

[2] K. Ramesh, A. S. Murthy, J. Senthilnath, and S. Omkar, "Automatic detection of powerlines in UAV remote sensed images," in *CATCON*. IEEE, 2015, pp. 17–21.

[3] T. Kersnovski, F. Gonzalez, and K. Morton, "A uav system for autonomous target detection and gas sensing," in *2017 IEEE aerospace conference*. IEEE, 2017, pp. 1–12.

[4] P. Grippa, "Decision making in a uav-based delivery system with impatient customers," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5034–5039.

[5] M. Ghazal, Y. Al Khalil, and H. Hajjdiab, "Uav-based remote sensing for vegetation cover estimation using ndvi imagery and level sets method," in *2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. IEEE, 2015, pp. 332–337.

[6] B. H. Y. Alsalam, K. Morton, D. Campbell, and F. Gonzalez, "Autonomous uav with vision based on-board decision making for remote sensing and precision agriculture," in *2017 IEEE Aerospace Conference*. IEEE, 2017, pp. 1–12.

[7] C. Holness, T. Matthews, K. Satchell, and E. C. Swindell, "Remote sensing archeological sites through unmanned aerial vehicle (uav) imaging," in *IGARSS*. IEEE, 2016, pp. 6695–6698.

[8] F. Heintz, P. Rudol, and P. Doherty, "From images to traffic behavior-a uav tracking and monitoring application," in *2007 10th International Conference on Information Fusion*. IEEE, 2007, pp. 1–8.

[9] P. E. Ross, "Open-source drones for fun and profit," *IEEE Spectrum*, vol. 51, no. 3, pp. 54–59, 2014.

[10] J. d. S. Barros, T. Oliveira, V. Nigam, and A. V. Brito, "A framework for the analysis of uav strategies using co-simulation," in *VI Brazilian Symposium on Computing Systems Engineering (SBESC)*, Nov 2016, pp. 9–15.

[11] J. de Sousa Barros, T. O. Freitas, V. Nigam, and A. V. Brito, "Analysis of design strategies for unmanned aerial vehicles using co-simulation," *Design Automation for Embedded Systems*, vol. 21, no. 3-4, pp. 157–172, 2017.

[12] A. V. Brito, A. V. Negreiros, C. Roth, O. Sander, and J. Becker, "Development and evaluation of distributed simulation of embedded systems using ptolemy and hla," in *Proceedings of the 2013 IEEE/ACM 17th International Symposium on Distributed Simulation and Real Time Applications*. IEEE Computer Society, 2013, pp. 189–196.

[13] (2018) Sitl simulator (software in the loop). [Online]. Available: http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html

[14] "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)– Federate Interface Specification," *IEEE Std 1516.1-2010 (Revision of IEEE Std 1516.1-2000)*, pp. 1–378, Aug 2010.

[15] J. J. Corner and G. B. Lamont, "Parallel simulation of uav swarm scenarios," in *Proceedings of the 36th Conference on Winter Simulation*, ser. WSC '04. Winter Simulation Conference, 2004, pp. 355–363.

[16] B. A. Kadrovach, "A communications modeling system for swarm-based sensors," Air Force Institute of Technology, Tech. Rep., 2003.

[17] Y. Wei, G. R. Madey, and M. B. Blake, "Agent-based simulation for uav swarm mission planning and execution," in *ADSS*, ser. ADSS 13. San Diego, CA, USA: Society for Computer Simulation International, 2013, pp. 2:1–2:8.

[18] N. R. Zema, A. Trotta, G. Sanahuja, E. Natalizio, M. D. Felice, and L. Bononi, "Cuscus: Communications-control distributed simulator," in *CCNC*, Jan 2017.

[19] W. G. La, S. Park, and H. Kim, "D-muns: Distributed multiple uavs' network simulator," in *ICUFN*. IEEE, 2017, pp. 15–17.

[20] I. A. Mason, V. Nigam, C. Talcott, and A. Brito, "A framework for analyzing adaptive autonomous aerial vehicles," in *International Conference on Software Engineering and Formal Methods*. Springer, 2017, pp. 406–422.

[21] M. Zhang, H. Qin, M. Lan, J. Lin, S. Wang, K. Liu, F. Lin, and B. M. Chen, "A high fidelity simulator for a quadrotor uav using ros and gazebo," in *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2015, pp. 002 846–002 851.

[22] C. Ptolemaeus, Ed., *System Design, Modeling, and Simulation using Ptolemy II*. Ptolemy.org, 2014. [Online]. Available: http://ptolemy.org/books/Systems

[23] C. Gomes, C. Thule, D. Broman, P. G. Larsen, and H. Vangheluwe, "Co-simulation: State of the art," *arXiv preprint arXiv:1702.00686*, 2017.

[24] S. I. S. Committee *et al.*, "Ieee standard for modeling and simulation (m & s) high level architecture (hla)-framework and rules," *pp. i–22*, 2000.

[25] IEEE, "Ieee standard for modeling and simulation (m&s) high level architecture (hla)—object model template (omt) specification," *IEEE No. 1516.2-2010*, 2010.