A Setup for Multi-UAV Hardware-in-the-Loop Simulations

Marcin Odelga, Paolo Stegagno, Heinrich H. Bülthoff and Aamir Ahmad

Abstract— **In this paper, we present a hardware-in-the-loop simulation setup for multi-UAV systems. With our setup we are able to command the robots simulated in Gazebo, a popular open source ROS-enabled physical simulator, using computational units that are embedded on our quadrotor UAVs. Hence, we can test in simulation not only the correct execution of algorithms, but also the computational feasibility directly on the robot's hardware. In addition, since our setup is inherently multi-robot, we can also test the communication flow among the robots. We provide two use cases to show the characteristics of our setup.**

I. INTRODUCTION

Latest trends in research on Unmanned Aerial Vehicles (UAVs) push for on-board integration of highly informative sensors such as laser scanners [1], cameras [2] and RGB-D devices [3], [4]. The increase in available information and the need to process it on-board requires also more computational power directly embedded on the robots. However, due to payload and power consumption constraints, the on-board available computational power is not yet comparable to the one of a normal desktop PC.

Another increasing trend is the use of physical simulations to test algorithms for robotics before the real hardware implementation phase. They are particularly useful when considering UAVs, since each experiment can be time consuming and could even result in a crash. However, whenever porting an algorithm from simulation to the real hardware, temporization issues may arise due to the limited on-board computational power. The main problem is that simulations usually run on a different hardware than the one equipped on-board, hence not allowing to check the real execution time of the software.

Hardware-in-the-loop (HIL) simulation [5] is a good way to test these aspects without a need of real robot experiments. In [6] the authors present a UAV system with HIL simulation for testing the platform with real-time data and real environment. Their system constitutes of a reliable platform for testing critical safety properties with special attention. Thus, through HIL simulation they greatly reduce experimental costs. Another HIL setup, for a UAV helicopter, can be seen in [7]. The authors show its cost-efficiency in terms of verification of the overall control performance and safety. Their system, capable of simulating flight tests including, e.g., basic flight motions and full-envelope flights, confirms the high effectiveness and usefulness of HIL simulations.

In this paper, we present our setup for HIL simulations which consist of two main parts. The first is Gazebo, a popular open source ROS-enabled simulator, which provides the dynamical simulation of one ore more UAVs and the corresponding sensor readings (IMU, cameras, etc.). On the other hand, each of the simulated UAVs is driven using an ARM-based Odroid board, which is the high level control board installed on our quadrotors. The interfacing between the components is provided by a ROS (Robot Operating System) node based on the Telekyb software ([8]).

Using this simulation scheme, we obtain two major benefits with respect to a normal simulation. First, we can test algorithms directly on the on-board computational units, hence also testing the computational times and the feasibility in real-time. In addition, since our setup is meant for

All authors are with the Max Planck Institute for Biological Cybernetics, Department of Human Perception Cognition and Action, Spemannstrasse 38, 72076 Tübingen, Germany {pstegagno, modelga, hhb, aamir.ahmad}@tuebingen.mpg.de

multi-robot systems, we can also test the communication among multiple boards.

The rest of the paper is as follows. In Sect. II we describe our hardware platform. In Sect. III we explain in detail the software setup to perform hardware-in-the-loop simulations. In Sect. IV we show two case studies in which we demonstrate the feasibility of hardware-in-the-loop simulations in both single and multi-UAV systems, and Sect. V concludes the paper.

II. UAV PLATFORM

Our robotic setup is made of multiple UAVs: MK-Quadro quadrotors from MikroKopter. Each UAV consists of a frame with four 10 inch propellers powered by brushless motors, motor controllers and a flight controller board with an 8-bit microcontroller. The main board includes an inertial measurement unit (IMU), i.e., two 3-axis, 10-bit analog sensors: an accelerometer (with a range of ± 2 g) and a gyroscope (± 300 deg/s range), both read with an analog to digital converter. The board communicates with the brushless motor controllers through a standard I^2C bus and offers two serial connections with a 115 200 Bd baud rate.

The original firmware, which allows us to drive the quadcopter with a remote control, has been replaced with our own software that has new features and a different interface that allows us to control the robot through serial channels. The first channel is used to send attitude and thrust commands to the microcontroller at ~100 Hz, and to receive low-frequency data (~20 Hz) (i.e. battery level). The second channel, strictly unidirectional, is used to retrieve high frequency IMU readings at 500 Hz. The platform is powered by a 2600 mAh LiPo battery that provides approximately 10 min of flight.

In addition, we have equipped the system with Odroid-XU3, a double quad core[1] ARM microprocessor board that provides enough computational power to make the system independent of external computational units. High computational power to weight/size ratio and low cost make this board relatively popular among the robotics community. It enables the use of high computationally demanding algorithms, e.g., visual based odometry and mapping ([4], [2], [9]) or advanced control methods such as model predictive control ([10]), directly on the platform.

[1]four Cortex-A15 at 2.0 Ghz, four Cortex-A7 and Heterogeneous Multi-Processing (HMP) solution for tasks management

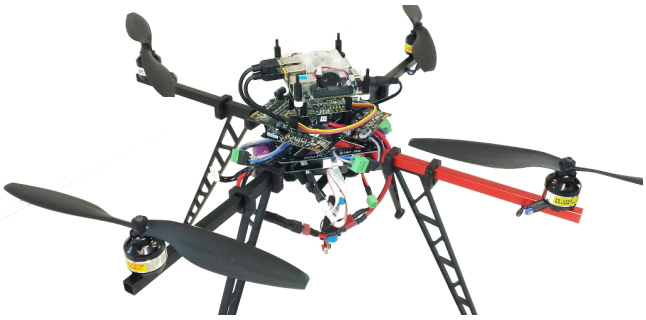

Fig. 1: One of our UAVs.

Communication with the low-level flight controller is carried out by two serial adapters connected to XU3's USB ports. Power to the board and its peripherals is provided by a 5 V step-down voltage regulator connected to the LiPo battery. The board can exchange data with the fixed operator desk using a USB Wi-Fi adapter. The complete platform is depicted in Fig. 1.

III. SOFTWARE SETUP

In this section, we describe first the software setup to drive a single UAV. Then, we show opportune modifications in order to run HIL simulations. Finally, we point out the required steps to perform multi-robot HIL simulations.

A. UAV control

In order to control a quadrotor, we use a software setup as depicted in Fig. 2. Being installed with the Ubuntu 14.04 ARM distribution, the Odroid board is ROS enabled, hence we can run the Telekyb software and its high-level controller and algorithms. Moreover, we can exploit ROS communication topics to connect it to a base station equipped with input/feedback devices. In addition, the base station may host appropriate routines to read measurements provided by a motion capture system (Vicon), if present, and translate them into ROS topics.

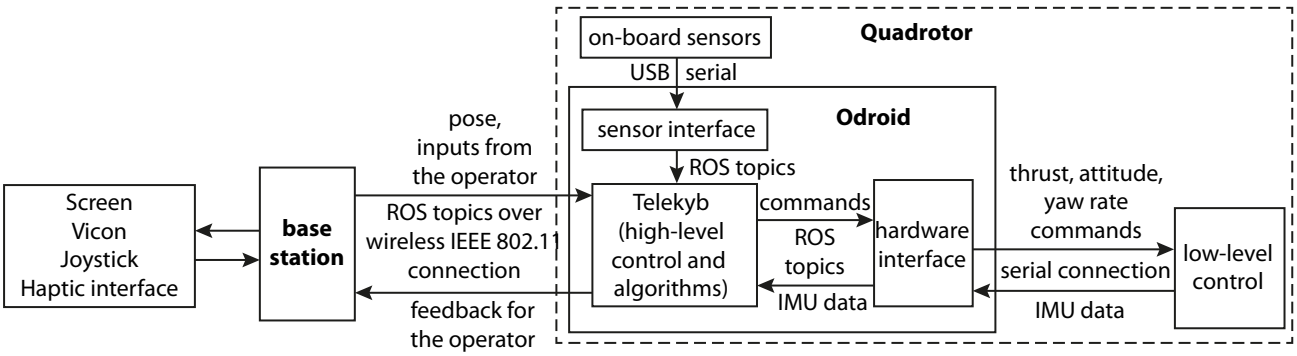

Fig. 2: A block scheme of our UAV setup.

Similarly, Telekyb modules are interfaced through ROS topics with a ROS node to connect it with the hardware. The role of this block is to encrypt and send the commands (desired thrust, attitude and yaw rate) to the microcontroller through the serial connection. The low-level controller is then in charge of driving the propellers' motors to follow the received commands. Similarly, the block receives IMU and battery status data from the microcontroller and translates them on ROS topics.

The UAV can be equipped with additional sensors such as cameras, RGB-D sensors, laser scanners, and GPS modules. In general, they can be connected through serial and USB ports which are present on the Odroid board. The interfacing with the high-level controller and algorithms is obtained once again with specific ROS nodes that translates the measurements into ROS topics.

B. Hardware-in-the-loop

In order to perform HIL simulations, we can exploit the standardization of the input/output provided by Telekyb, whose interfacing with other blocks is run solely through ROS topics. A block scheme of our setup to perform hardware-in-the-loop simulations is depicted in Fig. 3. With respect to the scheme in Fig. 2, the hardware and sensor interface block has been replaced with a Gazebo interface block. Its main functionality is to provide an interfacing layer between Gazebo and the high-level control and algorithms.

In particular, it uses data provided by the sensors and ground truth in Gazebo to emulate the ROS topics provided by the Vicon tracking system (if required), IMU, cameras and other sensors present both in the simulation and the real robot setup. The emulation not only comprises the specific topics and format on which the data are provided by Gazebo, but also take care of the temporization. For example, our real Vicon system provides data at $120Hz$, while the physical simulation is performed with a timestep of $1ms$, so that ground truth pose information about the UAV is available at $1000Hz$. Nevertheless, the pose provided to the Telekyb blocks is temporized at $120Hz$ by the Gazebo interface.

On the other hand, the Gazebo interface translates the thrust, attitude and yaw rate commands provided by the high-level controller into commands that can be read from the simulator and applied to the simulated UAV model.

The interfacing with the base station and the operator does not change with respect to the real robot system. The Gazebo simulator can be hosted either on the base station or on another machine. In order to test also the communication link between the robot and the base station, the latter is preferred, while the first can be implemented if the objective of the simulation is only to test the functionality and the execution time of the implemented algorithms. In both cases, the communication between the Odroid board and Gazebo is achieved through ROS topics over IEEE 802.11 connection.

C. Multi-robot

Since all components of the hardware-in-the-loop simulation, and in particular Telekyb, are inherently thought for multi-robot applications, only

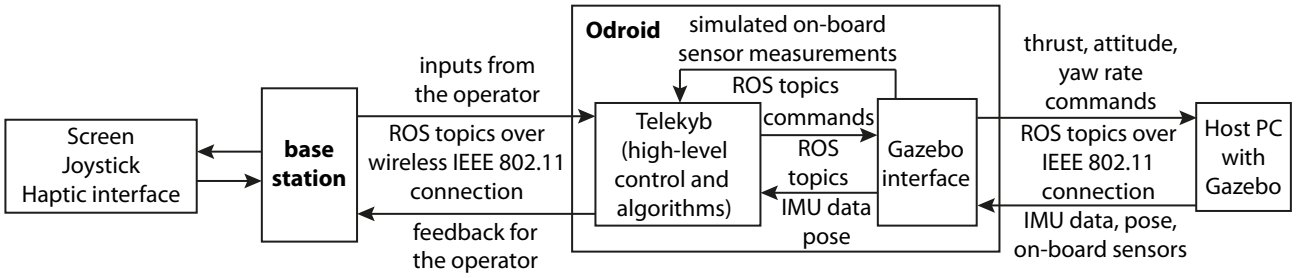

Fig. 3: A block scheme of the hardware-in-the-loop simulation setup.

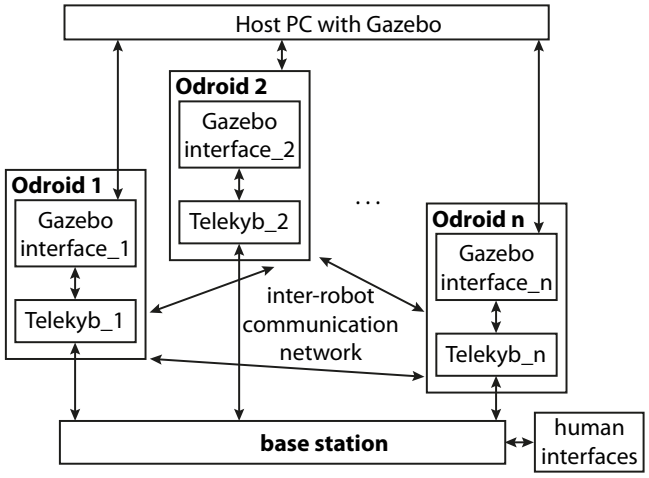

Fig. 4: A block scheme of the multi-robot hardware-in-the-loop simulation setup.

few adjustments are required to perform multi-robot HIL simulations.

The most important thing is that each robot simulated in Gazebo is endowed with a unique identifier ID, which must be inserted in the name of all ROS nodes and topics relative to such robot. Hence, for each robot in Gazebo, one Odroid board is setup to host a Gazebo interface that reads the appropriate topics (i.e.: the topics containing the own ID) and a Telekyb instance connected to such ID. Each Odroid is also connected to the base station in the same way as the previous case. The resulting scheme is presented in Fig. 4.

In addition to the single robot case, it is also possible to test the inter-robot communication network. At the current stage of development, the communication between robots is yet again performed exploiting ROS topics on a wireless IEEE 802.11 channel. Nevertheless, it is also possible to include and test custom communication networks among the robots equipping Odroid boards with appropriate hardware (e.g.: bluetooth antennas).

IV. CASE STUDY

As examples of the functionalities of our hardware-in-the-loop setup, we present in this section two case studies. In the first case study, we will show a single robot obstacle avoidance simulation and the corresponding experiment with a real robot. The aim is to show the same behavior in these two systems, which proves the possibility to test complex algorithms in simulation.

In the second case study, we highlight the multi-robot capabilities of our hardware-in-the-loop simulation scheme by performing formation control with three simulated UAVs driven by three Odroid boards.

A. Obstacle Avoidance

In this case study, we show an obstacle avoidance experiment performed both in simulation and on a real robot. The algorithm is divided into two main modules. The first module maps the obstacles in the proximity of the robot detected using depth images from an on-board RGB-D sensor. The second module, the avoidance part, works on a shared control principle as the user input, the velocity commanded by the operator, is altered to prevent collisions. The algorithm can override commands by adding a lateral and/or vertical component. The resulting reference velocity is then provided to the velocity controller of the robot[2]. To facilitate the

[2]Further details of this method are not in the scope of this paper. A manuscript explaining this algorithm in detail is in the review process during the preparation of this paper.

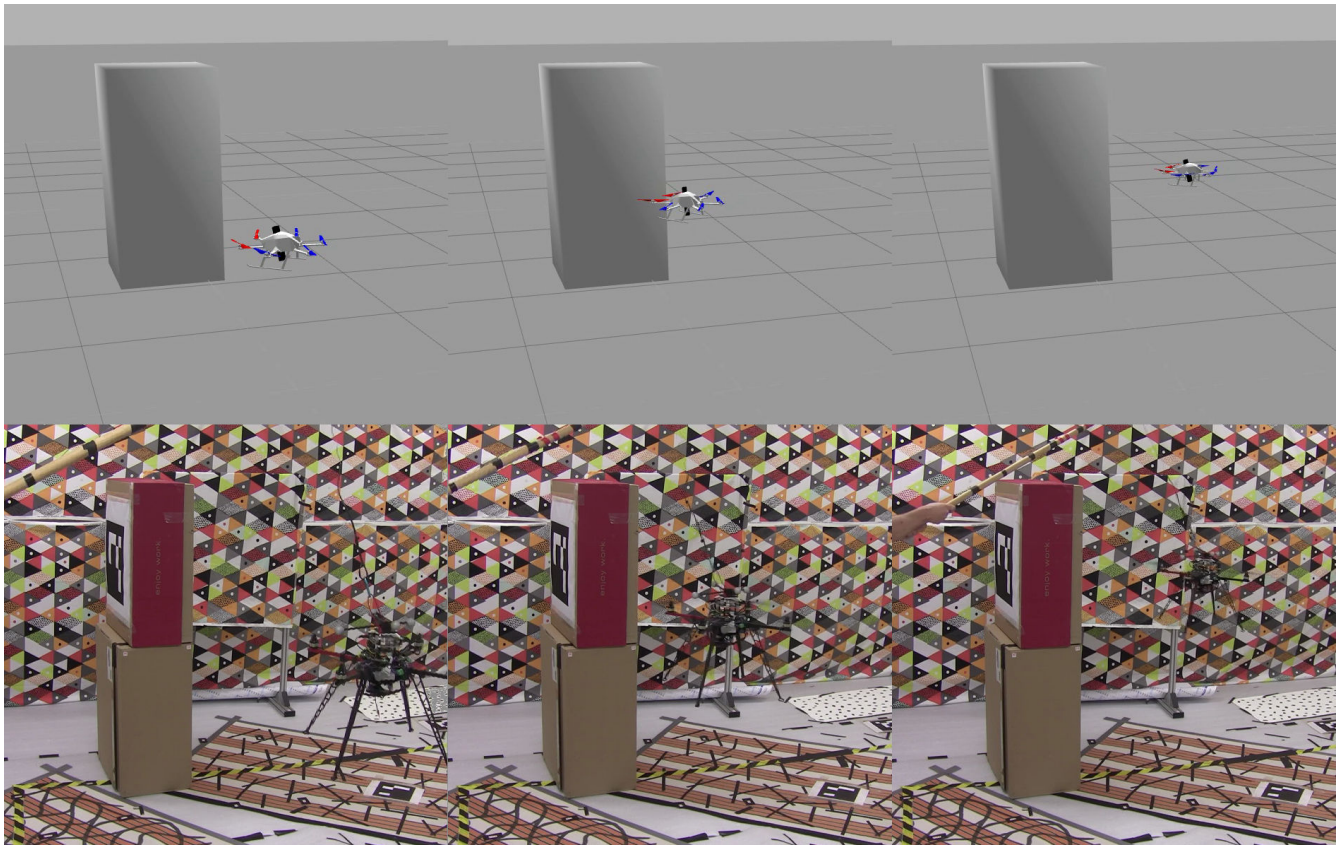

Fig. 5: Three comparative snapshots of the same obstacle avoidance algorithm performed in hardware-in-the-loop simulation (top) and a real UAV experiment (bottom).

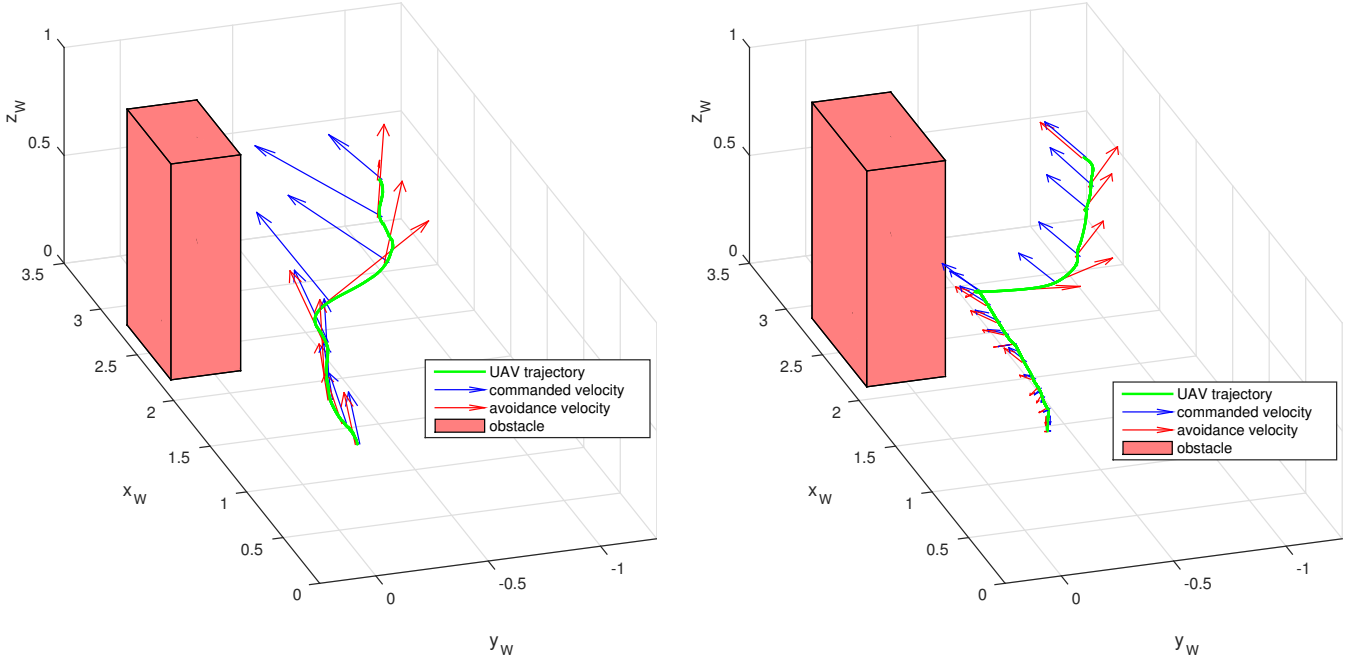

Fig. 6: The comparison of trajectories performed by the UAV in a real experiment (left) and in hardware-in-the-loop simulations (right).

teleoperation task the operator is provided with a visual and haptic feedback.

The on-board RGB-D sensor (Asus Xtion Pro), used as the source of data for obstacle detection, is mounted approximately 45° to the right with respect to the front propeller and rotated 90° around the camera axis to extend the vertical field of view (FOV). It is also rotated downward at about 20° to increase the number of visual features inside the FOV by framing a bigger portion of the ground, while simultaneously offering a horizontal line of sight to the operator. The Odroid board and the camera are attached rigidly to the frame with 3D printed parts. The UAV used in the real experiment and equipped with the RGB-D sensor weights approximately 1.3 kg.

In order to conduct the experiment with simulated UAV and sensor readings the same setup is recreated in Gazebo using a dynamic model of multirotor[3]. Color and depth images are generated thanks to an RGB-D sensor plug-in. The intrinsic and extrinsic parameters, i.e. camera parameters and sensor pose, respectively, are set to match the real hardware.

We have conducted experiments of lateral avoidance of a vertical obstacle, both in simulation and using the real robot. The human operator was commanding the robot to go towards the edge of the obstacle. In Fig. 5 snapshots from the experiments are presented and show consecutive positions of the robot during the flight. Full trajectories, together with the commanded and reference velocities, are presented in Fig. 6. Both Figures show that the real and the simulated robot have the same behavior in the experiment.

In Table I we show mean values (μ) and standard deviations (σ) of the execution time of two modules of the algorithm. These modules are related to the data processing of depth images and robot's state estimate, respectively. We have run the same experiment on three different platforms: standard desktop PC, hardware-in-the-loop simulation, and the real platform. Table I shows that the computational times obtained in the PC simulation is very different with respect to the timing obtained in the other two experiments. On the contrary, the computational times in the HIL simulation and in the experiment with a real robot are comparable. Small differences in the execution times in HIL and real experiments are due to the real sensor acquisition time, which is anyhow negligible compared to the rest of the algorithm's execution time. It is clear from the table that the timing obtained on the PC does not provide any meaningful insight into the evaluation of the real-time feasibility on the real hardware. However, using HIL simulation we were able to evaluate the real-time execution time which was comparable to the real experiment's execution time.

[3]https://github.com/ethz-asl/rotors_simulator

	Module 1		Module 2	
	μ $[ms]$	σ $[ms]$	μ $[ms]$	σ $[ms]$
PC	<1	-	<1	-
HIL	1.74	0.302	5.3	1.42
Robot	1.95	0.339	5.65	0.78

TABLE I: Execution Time Comparison

Fig. 7: A snapshot of formation control performed in hardware-in-the-loop-simulation.

B. Multirobot

In order to test the multi-robot capabilities of our HIL setup, we have performed a multi-robot teleoperation simulation with three simulated UAVs driven by three separate Odroid-XU3 boards following the scheme of Fig. 4. The navigation algorithm for the robots is the one proposed in [11].

The test algorithm was specifically chosen because it requires some exchange of information

among the robots in order to achieve consensus on the status of the system. This inter-robot communication was performed using the wireless IEEE 802.11 capabilities of the Odroid-XU3 boards. In Fig. 7 we show one snapshot of this simulation. The interested reader is invited to watch the video clip of this and the previous simulation and experiments in the accompanying multimedia attachment.

V. CONCLUSION

In this paper, we have presented a UAV hardware-in-the-loop simulation scheme which allows us to test the computational requirements of our algorithms directly on the computational unit that is equipped on the robots, while simultaneously enjoying the safety of a simulation. Additionally, we can perform multi-robot HIL simulations to test the communication among the robots.

The use of HIL simulations was very useful during the development of an obstacle avoidance algorithm, especially in terms of time needed to perform experiments. In fact, it was possible to test various parameters without the need to run real experiments. The use of the actual hardware (Odroid-XU3) in the simulations allowed to check if the execution times of the various modules were compatible with the on-line execution.

As future development, we plan to extend the system to include other types of robots (e.g.: wheeled robots) and test other types of communication networks (e.g.: bluetooth).

REFERENCES

[1] S. Omari, M.-D. Hua, G. Ducard, and T. Hamel, "Bilateral Haptic Teleoperation of an Industrial Multirotor UAV," in *Gearing up and accelerating cross-fertilization between academic and industrial robotics research in Europe*, Springer International Publishing, 2014, pp. 301-320.

[2] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-Direct Monocular Visual Odometry," in IEEE International Conference on Robotics and Automation (ICRA), 2014.

[3] P. Stegagno, M. Basile, H. H. Büthoff, and A. Franchi, "A Semi-autonomous UAV Platform for Indoor Remote Operation with Visual and Haptic Feedback," in IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 3862-3869.

[4] Z. Fang and S. Scherer, "Real-time Onboard 6DoF Localization of an Indoor MAV in Degraded Visual Environments Using a RGB-D Camera," in IEEE International Conference on Robotics and Automation (ICRA), May 2015.

[5] J. Burbank, W. Kasch, and J. Ward, "Hardware-in-the-Loop Simulations," in *An Introduction to Network Modeling and Simulation for the Practicing Engineer*, 1, Wiley-IEEE Press, 2011, pp.114-142.

[6] V.K. Chandhrasekaran and Choi Eunmi, "Fault tolerance system for UAV using hardware in the Loop Simulation," in International Conference on New Trends in Information Science and Service Science (NISS), 11-13 May 2010, pp.293-300.

[7] G. Cai, B. M. Chen, T. H. Lee, and M. Dong, "Design and implementation of a hardware-in-the-loop simulation system for small-scale UAV helicopters," in IEEE International Conference on Automation and Logistics (ICAL), 1-3 Sept. 2008, pp.29-34.

[8] V. Grabe, M. Riedel, H. H. Bülthoff, P. R. Giordano, and A. Franchi, "The TeleKyb framework for a modular and extendible ROS-based quadrotor control," in European Conference on Mobile Robots (ECMR), 25-27 Sept. 2013., pp.19-25.

[9] G. Mohanarajah, V. Usenko, M. Singh, M. Waibel, and R. D'Andrea, "Cloud-based collaborative 3D mapping in real-time with low-cost robots," in IEEE Transactions on Automation Science and Engineering, March 2014.

[10] Y. Liu, J. M. Montenbruck, P. Stegagno, F. Allgöwer, A. Zell, "A Robust Nonlinear Controller for Nontrivial Quadrotor Maneuvers: Approach and Verification", in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 28th Sep-3rd Oct 2015.

[11] A. Franchi, C. Secchi, H. I. Son, H. H. Bülthoff and P. Robuffo Giordano, "Bilateral Teleoperation of Groups of Mobile Robots with Time-Varying Topology" in IEEE Transaction on Robotics, 28(5) pp. 1019-1033, Oct 2012.