A flexible hardware-in-the-loop architecture for UAVs

Peter Lepej, Angel Santamaria-Navarro and Joan Solà

***Abstract*— As robotic technology matures, fully autonomous robots become a realistic possibility, but demand very complex solutions to be rapidly engineered. In order to be able to quickly set up a working autonomous system, and to reduce the gap between simulated and real experiments, we propose a modular, upgradeable and flexible hardware-in-the-loop (HIL) architecture, which hybridizes the simulated and real settings. We take as use case the autonomous exploration of dense forests with UAVs, with the aim of creating useful maps for forest inspection, cataloging, or to compute other metrics such as total wood volume. As the first step in the development of the full system, in this paper we implement a fraction of this architecture, comprising assisted localization, and automatic methods for mapping, planning and motion execution. Specifically we are able to simulate the use of a 3D LIDAR endowed below an actual UAV autonomously navigating among simulated obstacles, thus the platform safety is not compromised. The full system is modular and takes profit of pieces either publicly available or easily programmed. We highlight the flexibility of the proposed HIL architecture to rapidly configure different experimental setups with a UAV in challenging terrain. Moreover, it can be extended to other robotic fields without further design. The HIL system uses the multi-platform ROS capabilities and only needs a motion capture system as external extra hardware, which is becoming standard equipment in all research labs dealing with mobile robots.**

I. Introduction

Thanks to an acceleration of technological solutions for robotics, robotic systems are becoming more and more mature. This derives in systems with a high degree of complexity, in which robots are asked to fulfill a wide range of functionalities in a coherent, robust, and efficient way. Setting up a fully autonomous system including sensing, environment modeling, planning, and execution through control laws, becomes a delicate task of trial and error, tuning, and redesign considerations. This is especially acute in naturally unstable robots, such as drones or humanoids, for which failures or imbalances in the system can derive in fatal crashes that may put people and expensive hardware at risk. Moreover, operation of such robots in poorly structured and evolving scenarios, such as busy factories or dense forests, increases the need for robustness up to levels far beyond those typically encountered in research labs. Early experimentation through simulation is a powerful way of doing a number of these trials without risk, but simulation often lies too far from the real world scenarios [1]. Hardware-in-the-loop (HIL) schemes come handy for narrowing this gap.

This paper describes a HIL architecture as an efficient approach to pursue safe research and development of a complete and complex robotic system. This work takes as a use case the problem of setting up a fully autonomous unmanned aerial vehicle (UAV) for automatic exploration of forests from their inside. Dense forests are very challenging scenarios for the deployment of UAVs: GPS signals are denied or corrupted, the areas to cover are huge, which demands flights to be planned and executed at agile speeds, and they constitute unstructured and evolving 3D scenarios with difficult lightning, and potentially very small obstacles that will be detected only at the last moment. Ideally, the user should specify only the area to cover by the UAV, and the UAV should start the flight and return with a full map of the forest, providing precious data for its cataloging, exploitation and/or conservation.

In order to have the complete system rapidly working and easily upgradable by just upgrading the parts, our architecture takes profit of pieces of software publicly available, and organizes them in an operative system. Our focus has been in creating a HIL research and development platform, which gives us the flexibility to experiment safely, regardless of the system being somewhere midway in the process of development. Summarizing, the idea is to create a full working system quickly, which starts simple but safe thanks to the HIL architecture, and to upgrade it with more and better features as time, resources and/or knowledge become available. Among others, our HIL scheme helps in three main aspects:

- We can **simulate the sensing part**. This allows us to try different sensor configurations without actually having to acquire and install them, and to decide on the best sensing solution. Then, the payload and associated dynamics can be analyzed and the UAV can be properly optimized for weight, power consumption, speed, etc.
- We can create any kind of **simulated scenario** (and perceive it with the simulated sensors) and deploy the actual UAV on it. For example, we can try different tree geometries, consider small moving twigs and leaves, add low bushes, uneven terrain, etc.
- HIL allows us to perform **safer real experiments**, removing real obstacles and substituting them by simulated ones, while using the real platform, with the sensor on it, hence with the correct dynamics, but substituting the real sensor data by simulated sensor data.

P. Lepej is with University of Maribor, Slovenia, peter.lepej@uni-mb.si.

A. Santamaria-Navarro and J. Solà are with the Institut de Robòtica i Informàtica Industrial, CSIC-UPC, Llorens Artigas 4-6, Barcelona 08028, Spain, asantamaria, jsola@iri.upc.edu.

This work has been funded by the EU project AEROARMS H2020-ICT-2014-1-644271.

The paper has supplementary multimedia material available at www.iri.upc.edu/people/asantamaria/files/hil.mp4.

The key of a HIL system is to hybridize simulation with real scenario parts. In [2], [3], [4] similar HIL schemes were presented. However, only the onboard autopilots are used as hardware, and the platform dynamics, the actuation, and the Earth's physical effects are simulated. In [5] an HIL scheme is presented to simulate the visual servo control performed by a fixed-wing aircraft. And similarly, [6] proposes a test platform to develop educational and research autopilot control systems. Similar approaches are [7] and [8], where a real UAV is stabilized using computer vision. The actual vehicle is controlled inside a lowspeed wind tunnel and its attitude estimation is used to obtain images from a virtual camera in a simulated scenario.

The navigation system of a UAV comprises localization, mapping, motion planning, motion generation and control, in a fully closed-loop manner. In a complete navigation system, suited for unstructured terrains, the localization and mapping tasks should be based on a simultaneous localization and mapping system (SLAM) such as [9], [10], [11]. Although the full HIL architecture is described in the following, we considered the SLAM module out of the paper scope to simplify the initial stages of the framework development and to allow easy and safe inlab experimentation. Without loss of generality, we substituted this module by a motion capture system with the hope that SLAM can be incorporate in further stages.

This document is structured as follows. In the next section we present a brief description of the relevant HIL architecture of the system, including its main building blocks. Section III goes over the simulated results and real experiments. Conclusions and future work are provided in Section IV.

II. System Setup

A. Flexible Hardware-in-the-Loop System

We propose a hardware-in-the-loop architecture that allows for a smooth transition from a fully simulated system to a fully autonomous system, incorporating SLAM as an optional module. The architecture makes it easy to switch between five basic configurations, as described hereafter (Figs. 1(a) to 1(e)).

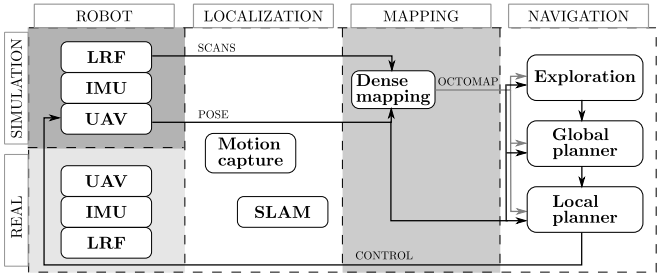
- **Simulation** (Fig. 1(a) and 1(b)). A UAV with a laser range finder (LRF) is deployed in a simulated environment. The pose of the vehicle can be obtained in a first stage from the simulator, Fig. 1(a), and then incorporate SLAM smoothly in a second stage, Fig. 1(b). This pose is used to stitch all scans together to form a well referenced Octomap ([12]). This scheme is useful to develop and validate the exploration, global planning and local planning techniques. Motion controllers close the loop back to the simulated UAV motors.
- **HIL system, with motion capture system** (Fig. 1(c)). A real UAV is deployed in a flying area, initially free of obstacles, provided with a motion capture system. The captured pose is used to simultaneously spawn a UAV in a simulated environment, with simulated obstacles, producing simulated sensor readings, *i.e.* laser scans. The captured pose is used to stitch all scans together to form a well referenced Octomap. Notice that the actual UAV can fly either in empty space (thus avoiding the risk of collisions), or in a cluttered space with obstacles accurately scanned and reproduced in the simulator (establishing this way a quasireal experiment). Motion controllers feed the real UAV motors.
- **HIL system, with SLAM** (Fig. 1(d)). SLAM is then incorporated in a new HIL system. The tracked UAV pose is used to simulate the scans in a simulated environment. Real IMU and simulated scans (and eventually other sensory information such as vision) are fused together in a SLAM module, thus obtaining an estimated pose, used with the scans to create dense Octomaps. The estimated and captured poses can be compared against each other for performance evaluation. Notice again that the real UAV can fly either in empty or cluttered space. When the system performs satisfactorily, we can switch to the fully autonomous configuration (next paragraph).
- **Fully autonomous system** (Fig. 1(e)) including full SLAM capabilities. The system is disconnected from any simulation or HIL modules. The motion capture unit can be used as a ground truth provider for the sake of performance evaluation. The validated system should be able to fly in other areas of similar complexity, away from the controlled lab conditions.

B. Navigation system

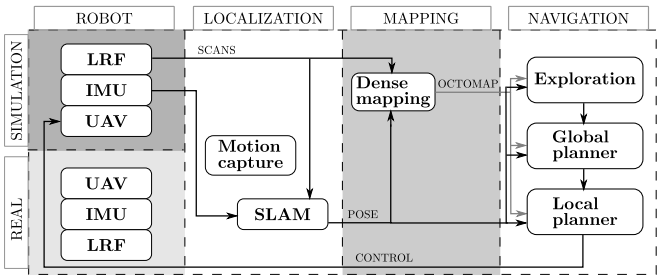
We have designed a modular system based on the Robot Operating System (ROS) framework. In this section we focus on four main blocks which together provide the desired navigation functionality: the localization and mapping tasks, the exploration technique, the global path planner (to reach the exploration goal) and the local path planner (which outputs the waypoint commands to the platform and contains the reactive behavior). All those modules are based on the current pose and the environment map. Localization comes from either simulation, motion capture, or SLAM, as explained in Section II-A. Dense mapping is produced by Octomap, an efficient 3D grid representation presented in [12], which takes profit of the precise localization to stitch all captured scans together into a full globally referenced map, called Octomap.

1) Exploration: Exploration is one of the elementary tasks in autonomous mobile robotics where the robot has to map the forest area in an efficient and fast manner. Several techniques have been presented to explore an unknown environment, and among them one of the simplest and functional is the frontier-based method. The basics of frontier-based exploration techniques were studied in [13], [14], [15], [16], [17]. Among these approaches, we have chosen the method which visits the closest frontier point. Its main operation can be described as follows.

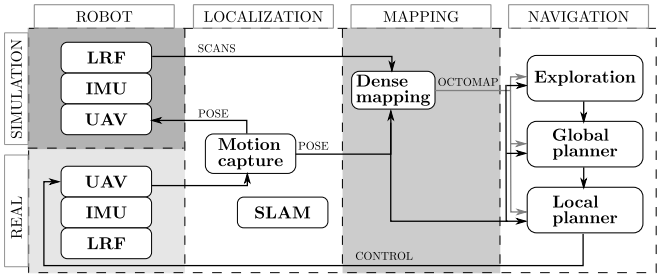
We define exploration rays from the robot's current pose to the current map border as shown in Fig. 2. The angle and distance steps for collision checks are tuning parameters that actually define the complexity and the performance



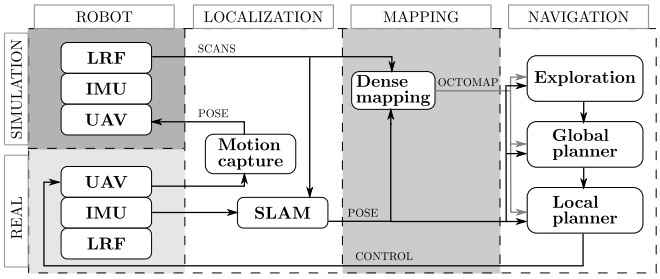
(a) Simulated system without SLAM.



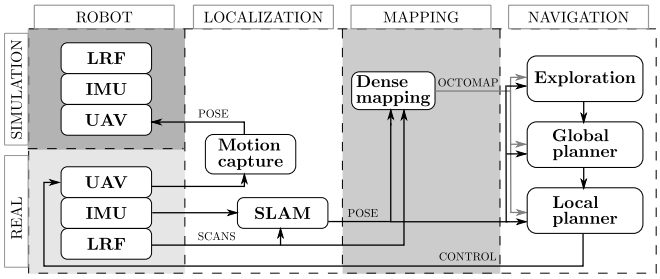
(b) Simulated system with SLAM.



(c) HIL system without SLAM.



(d) HIL system with SLAM.



(e) Full system with SLAM.

Fig. 1. Arquitecture configuration schemes.

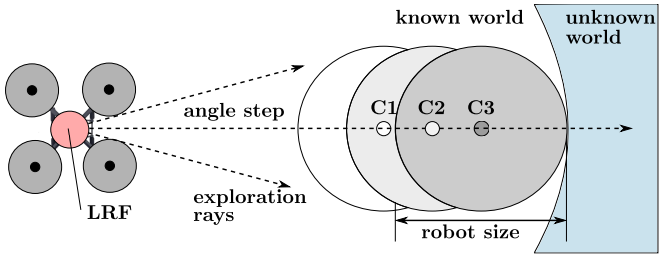


Fig. 2. Frontier-based exploration concept used in the UAV navigation system.

of the exploration algorithm. For each ray, we search for free spots among the candidates (*e.g.* C1, C2, C3) near the frontier between known and unknown terrain. If the space around the calculated position is free, which is defined by the robot size in a form of a cylinder, it is considered as a potential exploration goal candidate. The proposed algorithm selects the outmost candidate (C3) and stores it in a list. The list is initially filtered based on a given world size parameters, whose size defines the maximum exploration area with respect to the robot start position. We choose the closest exploration candidate from the list to establish the next exploration goal. We maintain the list of other potential goals in case the currently selected goal is not reachable.

2) Global Planner: The global path planner tries to compute a free path from the robot to the desired exploration goal based on the current pose and Octomap status. The resulting path is represented as a list of waypoints.

The planner is based on the Visibility Graph path search algorithm from chapter 6.2 in [13]. We have simplified it slightly in the sense that only one possible path around the detected obstacles is checked, so we have no guarantee of this path being the shortest one. However, given the typical shapes of obstacles in a forest (tree trunks), we found it reasonable to do it this way.

The global path planner starts by computing the straight path to the given goal, and checking for collisions on this path (Fig. 3). The collision checking is done considering the robot as a cylindrical bounding shape. If a collision is detected, the initial and final locations of the obstacle in the path line are used to create a collision bounding box (shown as an orange rectangular box in Fig. 3). The free space search or collision checking is done parallel to the longest side of the bounding box based on the limitation of Z axis (vertical) and offside distance parameter. Offside distance parameter is defined by the user and represent the search distance from the edge of the collision box.

The free positions found are stored in a list, from which we choose the best candidates to plan a collision free path around the collision box. First, we decide which is the most interesting side of the collision box based on the largest number of free poses found, and we keep the middle of all points on that side. We then check for the collisions on the resulting path which is now formed with all previously found free points. This is repeated until we have a free path

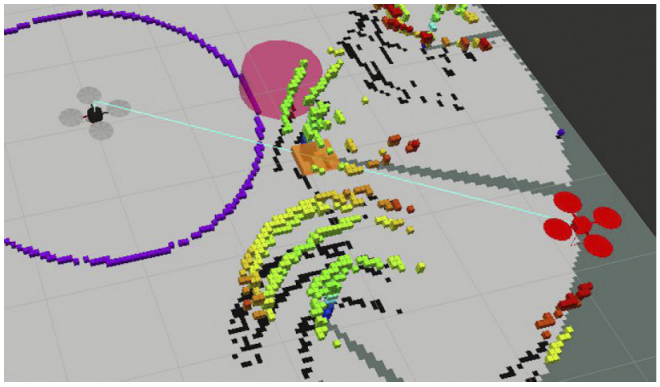


Fig. 3. The path planning algorithm shown in a 3D visualizer (ROS rviz): the red UAV model represents the goal position; the blue straight line represents the shortest path to the goal; the orange square box represents a collision box; and the red cylinder shape shows an example of a free space search around the collision box.

to the given goal. If a free path cannot be found we recall the global goals list from the Exploration module for a new goal candidate.

3) Local Planner: The local planner includes a reactive behavior, necessary to safely navigate in the dense forest environment. This topic has been studied in [18], [19] and [20]. The reactive behavior is designed to find a safe way, even if new obstacles have appeared in the environment as the result of an updated Octomap.

The reactive part of the local planner is done based on a cylinder shape whose dimension is defined by a lookahead distance parameter set by the user. Based on this cylindrical shape, we perform checking for the occupied cells in the Octomap. At maximum rate of 5 times per second, the Octomap provides an updated world model which triggers the reactive force calculations. If there are no occupied cells in the lookahead area, the local planner does not deform the global path. Otherwise, reactive vector forces, f, are computed from the average distance between the robot and all occupied cells that lie closer to the lookahead distance d, according to

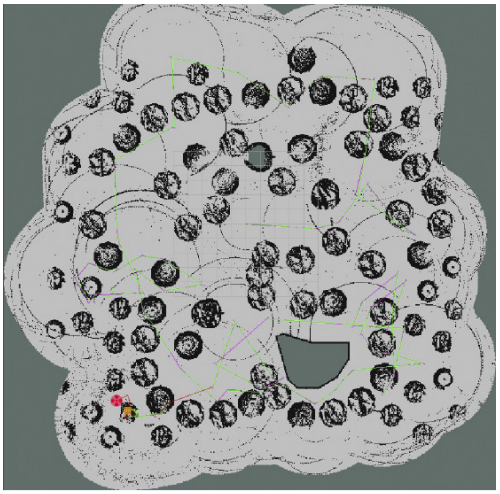
$$f = -u \frac{d}{\sum_{d_i<d} d_i}. \quad (1)$$

where u is a directed vector from the current waypoint X_c to the i-th Octomap's cell, and d_i is the distance to that cell. The local planner uses f to deform the current path's waypoints according to

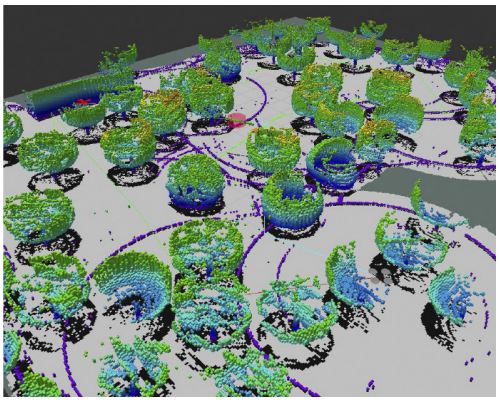
$$X_d = X_c + r\, f \quad (2)$$

where X_d is the deviated waypoint, and r is a tuning parameter that controls the effect of the reactive forces.

The produced waypoints are fed to the waypoints controller module. This controller resides in the embedded computer (on-board) and incorporates a platform states machine to deal with all platform modes, thus allowing only the execution of waypoints' control during flight mode (the takeoff and land will be performed internally in the states machine). Our implementation of the control law corresponds to a well known PID controller which produces velocity commands to achieve the next waypoint in the list. These velocity commands are fed to the attitude controller which actuates the motors to accomplish the desired motions.



(a) Top view where black circular elements are the treetops. The explored and unexplored zones are shown in clear and dark gray colors, respectively.



(b) Obtained 3D octomap.

Fig. 4. Results of the autonomous exploration in a simulated environment with several trees distributed along an area of 20x20m.

III. Experiments

The experiments corresponding to the schemes presented in Section II, comprising from simulation to different HIL implementations, are detailed hereafter and shown in the accompaining video.

A. Simulation

The simulation setup (architectures shown in Fig. 1) uses the Gazebo simulator[1] as the main physics machine. We take advantage of the Hector quadrotor gazebo plugins[2], modified to match the real quadrotor specifications, described in the following section. We assume that inside a forest there will

[1]http://gazebosim.org

[2]http://wiki.ros.org/hector_quadrotor

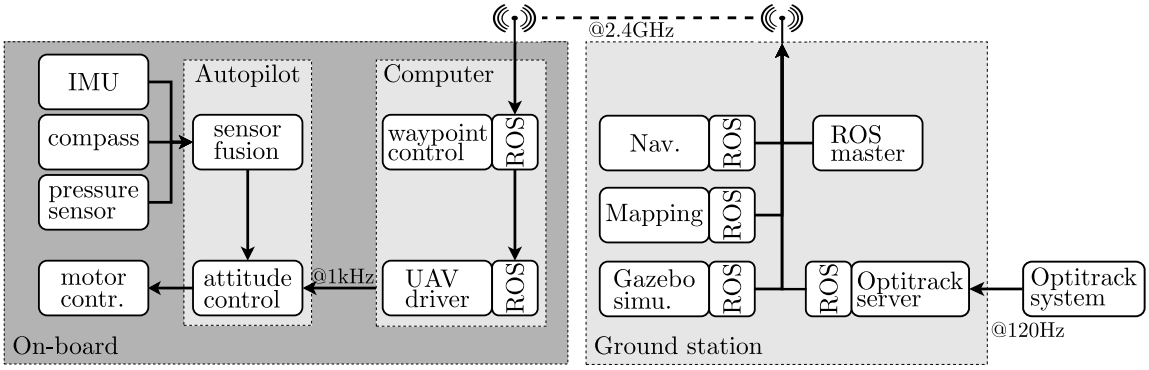


Fig. 5. Hardware setup including main processors, functionalities and communications.

be null presence or very small amount of wind. We simulate a set of trees distributed over a flat ground area and to reduce the computational burden of the simulation. Without loss of generality, we limited the simulated area to 20×20m. This first simulation step is crucial to test all algorithms without compromising the real platform.

Fig. 4 shows the 3D map resulting of applying the exploration, navigation, mapping and control techniques mentioned in previous sections. Notice how in Fig. 4 not only the tree trunks are present in the 3D Octomap, but also some important parts of the treetops.

B. Hardware-in-the-Loop

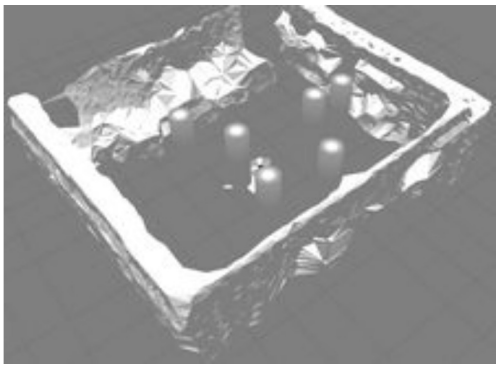
The robot used in the real experiments is based on an ASCTEC Pelican quadrotor[3], and the hardware setup is composed of two main systems, as shown in Fig. 5, including the ground station (Intel Core i7, 8 Gb RAM) and the onboard processors (*i.e.* an embedded Intel Atom @1.6GHz, 1Gb RAM, and the ASCTEC autopilot running the attitude controller).

In order to work with the virtual and real testbeds at the same time, we prepared a 3D reconstruction of the laboratory space (empty of obstacles) where a motion capture system (Optitrack) has his workspace defined. A dense point cloud was gathered using a custom-built 3D laser with a Hokuyo UTM30LX scanner mounted in a slipring. Each scan has 194,580 points with a resolution of 0.5 deg azimuth and 0.25 deg elevation. Then, we created the 3D model and imported it to the Gazebo simulator. Although the simulated environment allows also simpler environment models, this procedure was done in order to have a scenario as real as possible.

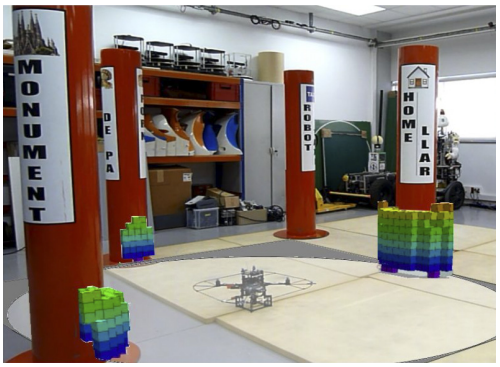
Notice how in the 3D virtual environment we are able to add virtual objects, see the vertical columns added in Fig. 6(a), or even simulate sensors exactly in the same pose as if they were attach to the real robot, such as a Velodyne VLP16. In the latter, the virtual sensor takes scans of the virtual scenario, which is a model of the real environment, defining what we call a quasireal environment, see Fig. 6(b).

Therefore, this 3D virtual world model can be used to deploy either a simulated UAV, corresponding to the scheme

[3]http://wiki.iri.upc.edu/index.php/Kinton



(a) Simulated environment composed by the 3D model of the real scenario together with virtual pillars emulating tree trunks.



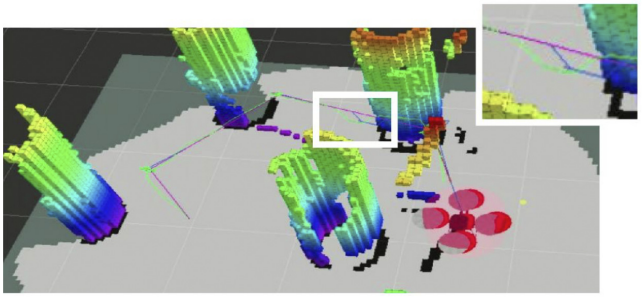
(b) Overlapping of the real and simulated environments with the Octomap containing the detected obstacles (coloured Voxels) and the sensing area (white circle with an alpha channel reduction).

Fig. 6. Real and Simulated scenario with vertical pillars to emulate tree trunks. Note how the 3D LIDAR of the experiments exists only in simulation but its readings are the used by the actual platform to navigate and map the obstacles.

in Fig. 1(a), or a real UAV, Fig. 1(c), creating a quasi-real experiment. In this second case, and during the first flights, we can remove the real obstacles thus achieving much safer paths for flight trials. Once the UAV closed-loop operation is judged robust enough, we can re-introduce the real obstacles.

Finally, Fig. 7 shows the resulting visualization of navigating in the real scenario with the quadrotor amongst the

real obstacles but using the virtual sensed pillars with the simulated 3D laser scan (Fig. 1(c)).



(a) Simulated environment.

(b) Real environment.

Fig. 7. Robot Kinton, navigating in real experiment with simulated pillars. The green path present robot trajectory, the purple path is the planned path and the blue path is the recorded reactive path.

IV. Conclusions

We have presented a modular, upgradeable and flexible hardware in the loop (HIL) architecture for research and development of fully autonomous robotic systems, especially indicated for UAVs deployed in challenging environments. We used as use-case the autonomous exploration of dense forests with UAVs. We have implemented the first steps of this architecture, and shown the potential of HIL systems to rapidly set and experiment with complex robotic systems. The full system is modular and takes profit of pieces either publicly available or easily programmed. The full HIL system uses the multiplatform ROS capabilities and only needs a motion capture system as external extra hardware, which is becoming standard equipment in all research labs dealing with UAVs. In the near future we plan to incorporate a SLAM pose estimation algorithm in the simulated, HIL and real environments, and to test the system in a real forest.

References

[1] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 3, Sept 2004, pp. 2149–2154 vol.3.

[2] D. Jung, J. Ratti, and P. Tsiotras, "Real-time implementation and validation of a new hierarchical path planning scheme of uavs via hardware-in-the-loop simulation," in *Unmanned Aircraft Systems*, K. Valavanis, P. Oh, and L. Piegl, Eds. Springer Netherlands, 2009, pp. 163–181.

[3] E. Mueller, *Hardware-in-the-loop Simulation Design for Evaluation of Unmanned Aerial Vehicle Control Systems*. American Institute of Aeronautics and Astronautics, 2007.

[4] 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," *Mechatronics*, vol. 19, no. 7, pp. 1057–1066, Oct. 2009.

[5] Y. A. Prabowo, B. R. Trilaksono, and F. R. Triputra, "Hardware in-the-loop simulation for visual servoing of fixed wing uav," in *Electrical Engineering and Informatics (ICEEI), 2015 International Conference on*, Aug 2015, pp. 247–252.

[6] Sufendi, B. R. Trilaksono, S. H. Nasution, and E. B. Purwanto, "Design and implementation of hardware-in-the-loop-simulation for uav using pid control method," in *Instrumentation, Communications, Information Technology, and Biomedical Engineering (ICICI-BME), 2013 3rd International Conference on*, Nov 2013, pp. 124–130.

[7] J. W. Grzywna, A. Jain, J. Plew, and M. C. Nechyba, "Rapid development of vision-based control for mavs through a virtual flight testbed," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, April 2005, pp. 3696–3702.

[8] N. Gans, W. Dixon, R. Lind, and A. Kurdila, "A hardware in the loop simulation platform for vision-based control of unmanned air vehicles," *Mechatronics*, vol. 19, no. 7, pp. 1043–1056, 2009.

[9] M. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (slam) problem," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 3, pp. 229–241, 2001.

[10] V. Ila, J. M. Porta, and J. Andrade-Cetto, "Information-based compact pose slam," *Robotics, IEEE Transactions on*, vol. 26, no. 1, pp. 78–93, 2010.

[11] C. Roussillon, A. Gonzalez, J. Solà, J.-M. Codol, N. Mansard, S. Lacroix, and M. Devy, "Rt-slam: a generic and real-time visual slam implementation," in *Computer Vision Systems*. Springer, 2011, pp. 31–40.

[12] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[13] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[14] C. Dornhege and A. Kleiner, "A frontier-void-based approach for autonomous exploration in 3d," *Advanced Robotics*, vol. 27, no. 6, pp. 459–468, 2013.

[15] K. Cesare, R. Skeele, S.-H. Yoo, Y. Zhang, and G. Hollinger, "Multi-uav exploration with limited communication and battery," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 2230–2235.

[16] C. Pravitra, G. Chowdhary, and E. Johnson, "A compact exploration strategy for indoor flight vehicles," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 3572–3577.

[17] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*. IEEE, 1997, pp. 146–151.

[18] A. A. A. Rizqi, A. I. Cahyadi, and T. B. Adji, "Path planning and formation control via potential function for uav quadrotor," in *Advanced Robotics and Intelligent Systems (ARIS), 2014 International Conference on*. IEEE, 2014, pp. 165–170.

[19] D. M. Rivera, F. A. Prieto, and R. Ramírez, "Trajectory planning for uavs in 3d environments using a moving band in potential sigmoid fields," in *Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012 Brazilian*. IEEE, 2012, pp. 115–119.

[20] A. Akhtar, S. L. Waslander, and C. Nielsen, "Path following for a quadrotor using dynamic extension and transverse feedback linearization," in *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 3551–3556.