Integrating Hardware-In-the-Loop Simulation and BIM for Planning UAV-based As-built MEP Inspection with Deep Learning Techniques

K. Wang[a] and J.C.P. Cheng[a]

[a]Department of Civil and Environmental Engineering, the Hong Kong University of Science and Technology, Hong Kong SAR
E-mail: kwangaw@connect.ust.hk, cejcheng@ust.hk

Abstract –

Unmanned Air Vehicles (UAVs) are increasingly used and have many potential applications in the architecture, engineering, and construction (AEC) industry. UAVs are now commonly controlled manually with an experienced survey engineer in field operation, but UAVs can also fly autonomously with the support of localization technologies such as simultaneous localization and mapping (SLAM). Autonomous flying of UAVs can reduce human effort and minimize human error. However, it requires careful flight path planning, active environment sensing, and effective obstacle avoidance techniques, especially where the built environment is complex and space is tight. Currently, algorithms flight path planning, object detection and obstacle avoidance are developed based on testing that involves actual Survey UAVs; this is not only time consuming but also expensive and hazardous in the sense that it may damage hardware and injury ground personnel if error-induced crashes occur. This paper presents a hardware-in-the-loop simulator which can simulate the behavior of a drone and generate synthetic images from the scene as datasets, detect and verify as-built MEP objects with a trained neural network, and generate point cloud data for validating as-designed BIM model with as-built BIM model in the future. The system architecture, operations, and performance of the developed simulator are discussed, followed by an illustrative example.

Keywords –

Automation; Flight Path Planning; Robotic Perception; Simulation; UAV

1 Introduction

Surveying using terrestrial laser scanners has been common practice in the AEC industry. Meanwhile, building information modeling [1] (BIM) has been increasingly required by both public and private sectors as a part of any building and infrastructure developments. As the more recent a design is, the higher level of details its digital 3D model will contain, and as-designed BIM models need to be validated against their as-built counterparts. For example, mechanical, electrical and plumbing (MEP) inspection of a tunnel traditionally requires inspectors to go inside the tunnel and use a terrestrial laser scanner to inspect and confirm coherence with the as-designed BIM model. This method normally requires much manpower, making it inefficient, time-consuming and risky. Some civil structures like tunnels and wastewater treatment plants can have hazardous volatile organic chemicals (VOCs) [2] which can lead to cancer and other medical problems. For another example, to conduct a bridge inspection, expensive high-energy laser scanners and a helicopter are often required. UAVs equipped with cameras, Lidar and/or other detection equipment are promising alternatives as they can collect point cloud data [3] and photogrammetry images [4] and generate 3D models for inspection and visualization.

Testing a physical survey UAV in a real-world environment can be dangerous, expensive and time-consuming. On April 28, 2017, the US Federal Aviation Administration (FAA) released a very detailed report on drone crash tests. The 107-page report was a joint research involving 23 academic institutions working together with the FAA with the aim to understand the risk and hazards (including possible injuries) of the system failure of a UAV. The objective is based on the FAA's research addressing seven questions: (1) What are the hazard severity criteria for a UAV collision (e.g. weight and kinetic energy)? (2) What is the severity of a UAV collision with aircraft on the ground? (3) What is the severity of a UAV collision with property on the ground? (4) What is the severity of a UAV collision with a person on the ground? (5) What are the characteristics of a UAV where it will not be a risk to an aircraft or person/property on the ground? (6) Can the severity of a UAV collision

with an aircraft or person/property on the ground be characterized into UAV categories and what would those categories look like? (7) How can UAV be designed so as to minimize the potential damage done during a collision? [5] In the study of human injuries due to loss of control of a UAV, Section 2.4.4 (Payload Loss Injury Applications) and Section 2.4.5 (Fire Injury Applications) of the report are the most relevant to our research interest: indoor UAV scanning application. In an indoor environment under the condition with no magnetic and wind interference, yet with low lighting and weak GPS signals, a UAV can still be prone to crashes even when the operator is an experienced pilot. Energy sources such as Lithium Polymer battery or Nitromethane mixed with methanol [6] (depending on the overall payload of the vehicle) may ignite upon ground collision, which poses danger to the ground operator. In addition, a survey UAV that carries high precision laser scanners, cameras, and/or various control and processing units on board are too expensive to be damaged. Even a single failure can result in a devastating crash. Therefore, designing a virtual environment close to the real environment for UAV simulation and training before testing a physical UAV on site is proposed in this framework.

Many simulators have been established in the fields of robotic research, computer vision, gaming technology. The UE4 [7] engine, for example, is considered a leading game engine among others such as Unity to provide many resources such as AAA-level realistic virtual environment, through its state-of-the-art shader and the recently improved ray tracing rendering technique. These photorealistic virtual environments can be used as a source for generating images for neural network training, such as training a convolutional neural network to detect objects. However, simulators like UE4 or Unity [8], despite the shader rich feature that they support, lack features such as predefined sensors models provided in the (robotics operating system) ROS [9] simulator.

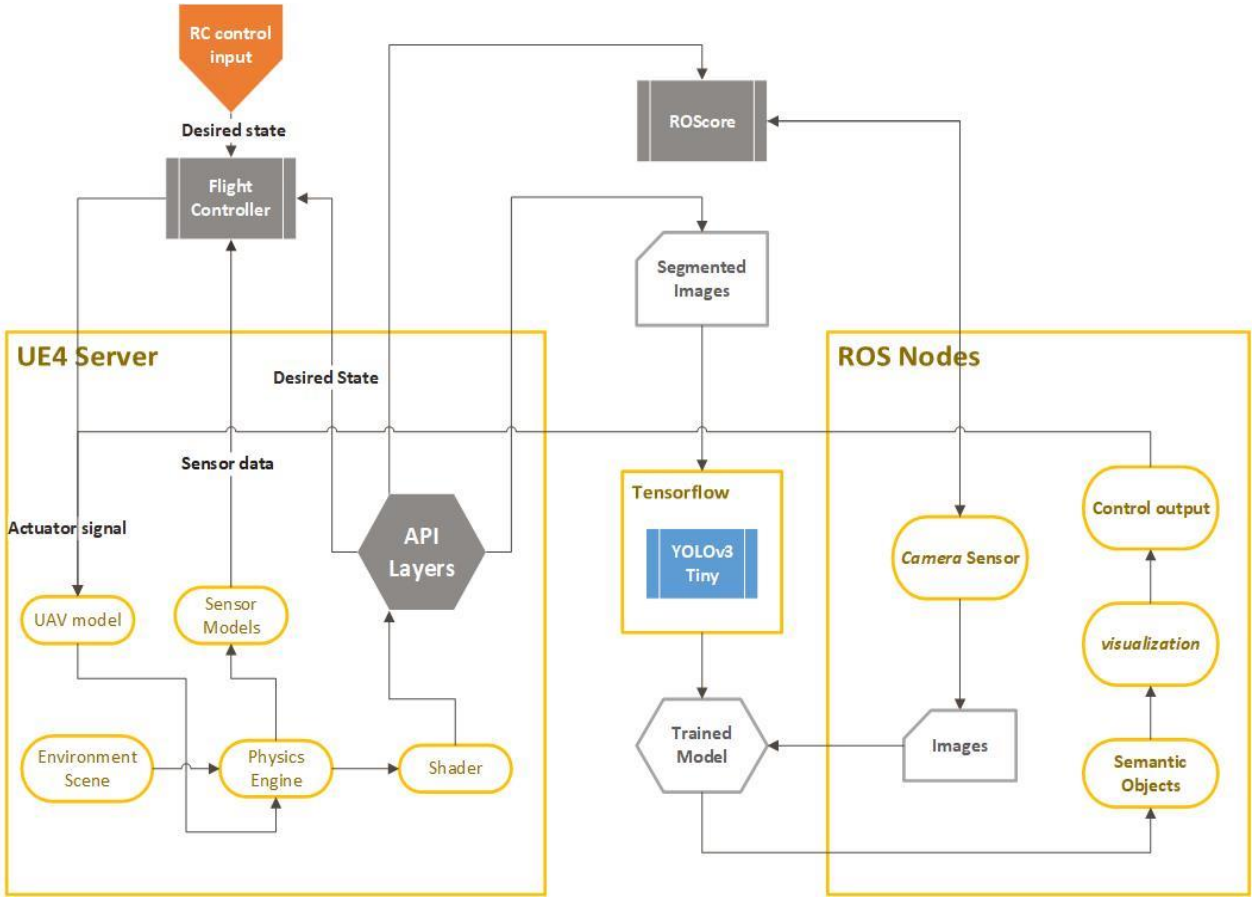

Figure 1. The developed framework for UAV planning using hardware-in-the-loop simulation and BIM technology

Section 2 illustrates the developed framework for UAV planning using hardware-in-the-loop simulation and BIM technology, which contains the following three major parts:

1. The creation of a high-fidelity simulation environment with MEP pipes models and connected flight controller in hardware-in-the-loop [10] mode.

2. The creation of ROS nodes that link the virtual Lidar from ROS to UE4 simulator and generate point cloud data and obstacle avoidance.

3. The creation of a perception system by training of a YOLO v3 [11] object detection neural network with pictures of MEP models.

Implementation details and results of these three parts will be presented separately in the followings.

2 Proposed Framework for UAV Planning Using Hardware-in-the-Loop Simulation and BIM

The proposed framework (as shown in Figure 1) starts where a radio control signal is passed from a radio controller to the virtual flight control module, which drives an actuator on the UAV. Gravity and weather information from the environment will then be sent to the physics engine to simulate the kinematics and return pose information of the UAV. The image API will then acquire high-fidelity images from the scene and compose a directory for neural network training, for the purpose of object detection. The ROS API layer will receive the vehicle pose and location information and via each ROS node for simultaneous localization and mapping, and trajectory planning.

2.1 Simulation Environment Setup

A MEP scene is focused in this study due to the complexity in geometry. The scene, which contains common MEP components such as pipes, elbows and supports, is generated in realistic visual effect using the UE4 engine. This virtual environment consists of MEP pipelines, both having similar object settings and made with appropriated UV to make it realistic. A virtual quadcopter is also created as an avatar to simulate a physical UAV with a weight of 3 kg. A physical flight controller is installed in hardware-in-the-loop mode, and we use a physical 9-channel radio controller which is plug in our server USB port to control the virtual quadcopter within the scene. The simulated environment is shown in Figure 2.

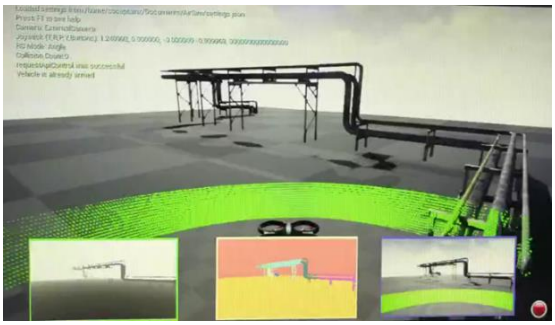

Figure 2. The simulation environment

2.2 Hardware-in-the-Loop simulation with Pixhawk Flight Controller

Hardware-in-the-loop simulation is a type of real-time simulation that are commonly used to test complex embedded system [12]. Figure 3 illustrated the relationship between the flight controller hardware and to the developed simulator.

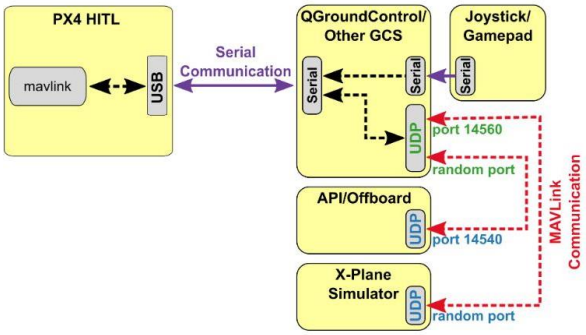

Figure 3. Connection between Pixhawk 4 and the developed simulator

Pixhawk 4 [13] is connected to the USB port of a server. Through the UDP port of the simulation engine, the flight controller can establish communication with the simulator and interact with the built-in physics sub-engine to send actuator signals, pass environment data (e.g. gravity, wind, atmospheric pressure, etc.), and update the real gyro and accelerometer sensors onboard the flight controller. In this way, the simulated environment can behave as close to the real environment as possible.

2.3 Training of an Object Detection Neural Network with YOLO v3 Tiny

The simulator has a Python API so that it can connect with Keras [14], a high-level deep learning

Python middleware that runs on top of Tensorflow [15], which we use for the neural network training (of xxx). There are numerous deep learning-based object detection architectures. YOLO (You Only Look Once) v3 Tiny [16] is used in this research because it is 100 times faster than Fast R-CNN [17] and it is a very small model, comprising only 9 convolutional layers, which is ideal where computational power is limited, such as what is available on the Nvidia TX2 [18] embedded unit. Darknet [19] is used as a trained deep architecture with trained YOLO weights. The training process consist of the following four steps.

1. Create a dataset which includes image classes of MEP models such as pipes, valves, industrial air conditioning units, fittings, water heating, etc. 2000 images are used as training set for each class.

2. Tag images via Visual Object Tagging Tool.

3. Batch and subdivision, using an input resolution of (416 x 416) with a batch of 64 and a subdivision of 3. (Note: values were chosen based on the maximum result achieved without CUDA [20] error.)

4. Train a model, weights will be given for every 100 epochs in Darknet. The result is shown in Table 1.

Table 1.

Input Resolution	416 x 416
Weights	YOLO v3-tiny-VM.cfg
Iterations	50,000
Average Loss	0.2103
Average IoU (%)	47.18%
mAP	62.39%

Finally, we will test the result of training by connecting this object detection network with our simulator via the Python API. MEP pipes are detected in the simulator as the simulated UAV flies, detects objects of interest, and marks their locations in the simulated scene in the virtual environment, as shown in Figure 4.

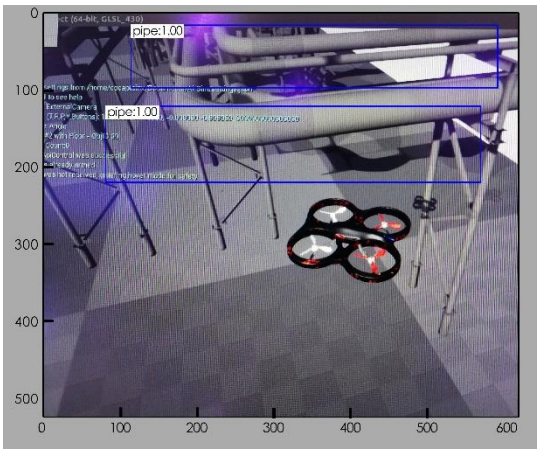

Figure 4. Detection with YOLO v3 Tiny

2.4 Generate Virtual Lidar Data with ROS and UE4

The virtual Lidar in the UE4 engine requires the implementation of two technologies: (1) ray tracing within the UE4 engine to have the visual effect of a laser beam, and (2) construction of virtual sensors within ROS via ROSbridge to publish the defined Lidar sensor reading from ROS to the listening port of the UE4 engine.

The parameters of the virtual Lidar in the simulator are configured in the same way as the RS-Lidar 32 with 32-channel laser, 640K points per second, +/- 15-degree FOV (vertical) and 360-degree FOV (horizontal). [21] The green line indicates the laser beam is generated, as shown in Figure 5 and Figure 6.

The acquired point cloud data will need to change the camera projection matrix with a Build Projection Matrix function, which computes a 4x4 matrix using FOV, width, height and near plane.

Figure 5. Virtual Lidar in the simulator

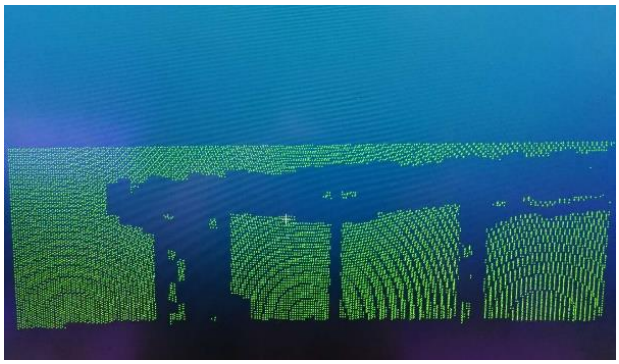

Figure 6. Point cloud visualization

3 Implementation and results

The proposed survey UAV is implemented with a DJI F350 chassis and a Nvidia TX2 embedded unit. The Nvidia Jetson TX2 features a Nvidia Denver dual-core and an ARM Cortex-A57 quad-core. TX2 also carries a 256-core Nvidia Pascal GPU with 8GB of RAM. The power consumption is only 7.5 watts. A stereo camera is incorporated for running a visual simultaneous localization and mapping (SLAM) [22] algorithm for the purpose of localization and obstacle avoidance. A 32-channel Lidar is also mounted on the system to generate some point cloud data. A Pixhawk flight controller with radio receiver is installed along with four electronic speed controllers and corresponding RC rotors. A 9-channel radio controller is used to control the UAV. The total weight of the system is approximately 3 kilograms with a 20K mAh Lithium Polymer battery.

Figure 6. UAV implementation

The stereo camera is a global shutter camera, which does not have an onboard IMU. It is calibrated using a checkerboard (shown in Figure 7) to calculate the intrinsic and extrinsic camera parameters with Pinhole model. With the calibrated stereo camera, we flew the UAV with a VINS SLAM algorithm running in TX2, and a circular trajectory is generated. The trajectory is visualized in Rviz as shown in Figure 8 and Figure 9.

Figure 7. Stereo Camera Calibration

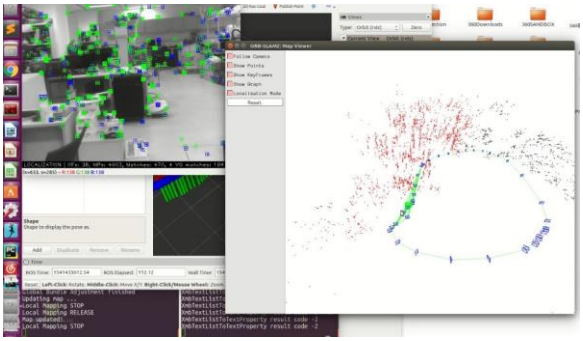

Figure 8. ORB SLAM in Rviz

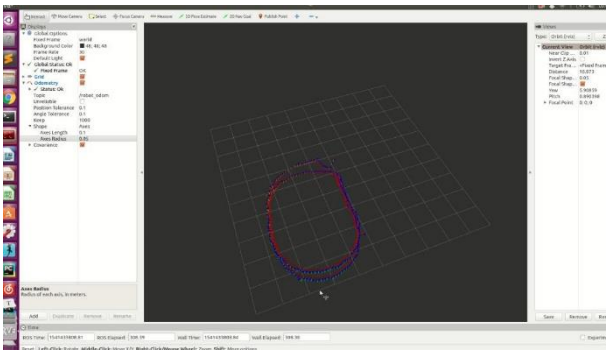

Figure 9. Trajectory in simulator

We also load the AirLib library from our simulator and deployed it on the TX2. The simulator uses the MavLinkCom component, which is a proxy architecture where a connection to the Pixhawk flight controller can be achieved via a serial or UDP port. The Pixhawk flight controller will send messages via MavLink, which allows the controller to be connected to the serial port of QGroundControl (QGC) [23]. We can control the UAV as a client to run on the TX2 meanwhile have it connected to our server with virtual simulation environment.

Running the VINS stereo algorithm along with the YOLO v3 Tiny object detection neural network gives a high load of computation. Usage statistics show that VINS uses roughly 60% of the processing power in the TX2.

4 Conclusion and Future Work

This paper presents a hardware-in-the-loop simulation, which can simulate the behavior of a UAV, generate synthetic images from the scene as datasets, detect and verify as-built MEP objects with a trained YOLO v3 neural network, and generate point cloud data for future as-designed BIM and as-built BIM validation. This demonstrates an effective way of testing the behavior of a UAV prior to live demo flight and blazes a trail toward implementing AI-based robotic inspectors in the AEC industry. Moreover, the capability of generating virtual point clouds in the simulation engine provides a low-cost way for verifying the trajectory of a survey UAV before actual UAV-based scanning takes place. This can simulate any missing point cloud region in the environment constructed with an as-designed BIM model.

The developed framework will be improved in the future. In particular, (1) the proposed framework has not been tested in a real industrial environment with MEP piping installed for field detection. Future tests at a plant room site with MEP components will be conducted. (2) A better stereo camera with IMU will be installed to give the VINs' SLAM performance a boost, potentially resulting in better localization and mapping. (3) A trajectory planning module will be developed on the system and further tested its stability with a motion capture system.

References

[1] Cheng, Jack CP, and Qiqi Lu. "A review of the efforts and roles of the public sector for BIM adoption worldwide." Journal of Information Technology in Construction (ITcon) 20.27 (2015): 442-478.

[2] Zhang, Qijun, et al. "Emission factors of volatile organic compounds (VOCs) based on the detailed vehicle classification in a tunnel study." Science of The Total Environment 624 (2018): 878-886.

[3] Wang, Qian, Jack Chin Pang Cheng, and Hoon Sohn. "Automatic Reconstruction of As-built BIM from Laser Scanned Data of Precast Concrete Elements for Dimensional Quality Assessment." Proceedings of the International Symposium on Automation and Robotics in Construction (ISARC). 2016.

[4] Colomina, Ismael, and Pere Molina. "Unmanned aerial systems for photogrammetry and remote sensing: A review." ISPRS Journal of photogrammetry and remote sensing 92 (2014): 79-97.

[5] Arterburn, David, et al. "FAA UAS Center of Excellence task A4: UAS ground collision severity evaluation, revision 2." (2017).

[6] Miller, Robert C., and Raymond M. Fuoss. "Electrolyte-Solvent Interaction. II. Quaternary Salts in Methanol-Nitromethane and Methanol-Benzene Mixtures." Journal of the American Chemical Society 75.13 (1953): 3076-3080.

[7] Qiu, Weichao, and Alan Yuille. "Unrealcv: Connecting computer vision to unreal engine." European Conference on Computer Vision. Springer, Cham, 2016.

[8] Haas, John K. "A history of the Unity game engine." (2014).

[9] Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

[10] Cai, Guowei, et al. "Design and implementation of a hardware-in-the-loop simulation system for small-scale UAV helicopters." Automation and Logistics, 2008. ICAL 2008. IEEE International Conference on. IEEE, 2008.

[11] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[12] McDonnell, Rachel, Martin Breidt, and Heinrich H. Bülthoff. "Render me real?: investigating the effect of render style on the perception of animated virtual humans." ACM Transactions on Graphics (TOG) 31.4 (2012): 91.

[13] Meier, Lorenz, et al. "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision." Autonomous Robots 33.1-2 (2012): 21-39.

[14] Chollet, François. "Keras: The python deep learning library." Astrophysics Source Code Library (2018).

[15] Abadi, Martín, et al. "Tensorflow: a system for large-scale machine learning." OSDI. Vol. 16. 2016.

[16] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint

arXiv:1804.02767 (2018).

[17] Girshick, Ross. "Fast r-cnn." Proceedings of the IEEE international conference on computer vision. 2015.

[18] NVIDIA, "NVIDIA Jetson TX2 Module" Online: https://developer.nvidia.com/embedded/buy/jetson-tx2, Accessed: 31/01/2019

[19] Shafiee, Mohammad Javad, et al. "Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video." arXiv preprint arXiv:1709.05943 (2017).

[20] Kirk, David. "NVIDIA CUDA software and GPU parallel computing architecture." ISMM. Vol. 7. 2007.

[21] Schwarz, Brent. "LIDAR: Mapping the world in 3D." Nature Photonics 4.7 (2010): 429.

[22] Dissanayake, MWM Gamini, et al. "A solution to the simultaneous localization and map building (SLAM) problem." IEEE Transactions on robotics and automation 17.3 (2001): 229-241.

[23] MEIER, Lorenz. "MAVLink Micro Air Vehicle Communication Protocol-QGroundControl GCS." QGroundControl GCS [online] (2009).