Software-in-the-Loop Simulation of UAS-UGV Cooperation in Precision Agriculture

Saron Bhoopathy[1], Kenta Hirashima[2], Varun Aggarwal[3], Aanis Ahmad[3], and Dharmendra Saraswat[4]

[1]School of Aeronautics and Astronautics, Purdue University, West Lafayette, IN 47907 USA
[2]Department of Mechanical Science and Engineering, University of Illinois Urbana-Champaign, Urbana, IL 61801 USA
[3]School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA
[4]Department of Agricultural and Biological Engineering, Purdue University, West Lafayette, IN 47907 USA

***Abstract*— With the growing human population and labor shortages during critical farm operations, modern farms are expected to maintain larger yields at lower costs despite facing reduced labor availability. Studies have shown potential for Unmanned Aerial Systems (UAS) and Unmanned Ground Vehicles (UGV) to cooperatively perform farm operations to help farmers address these challenging issues. However, a collaborative system capable of autonomous crop scouting is still lacking. Furthermore, testing software and algorithms in simulated environments before real-world implementation is needed to avoid damage to expensive equipment. This project aims to develop a simulation environment for evaluating cooperative algorithms for UAS-UGV systems in agriculture. The simulation environment was developed using Robot Operating System 2 (ROS2) software framework, PX4 flight controller firmware, Gazebo 11 simulator, and QGroundControl ground station software. Computer vision was enabled on UAS using the You Only Look Once (YOLO) object detection algorithm. Communication between UAS, UGV, and the ground station was implemented using the MAVLink communication protocol. Preliminary results indicate that a scouting UAS successfully sends location coordinates of a target of interest to UGV in a simulated corn field. Overall, the project provides a viable simulation of cooperative agricultural robots that other researchers can use to evaluate their systems. Further, using open-source packages ensure seamless deployment of software and algorithms to physical systems with minimal code changes.**

I. Introduction

In 2021, about 193 million people in 53 countries or territories experienced acute food insecurity. This represents an increase in 40 million people in comparison to already record numbers in 2020 [1]. According to the UN World Population Prospects [2], the global population is set to reach 8 billion people mid-November of 2022. Thereafter, the global population is estimated to rise over 10 billion by 2059. Therefore, farms will need to produce larger yields at higher quality to meet the nutritional needs of the population. Crop diseases and weeds pose a significant detriment to crop yield. It is reported that weeds lead to a 30% reduction in crop yield worldwide [3]. Global yield losses due to crop pests and diseases on food crops range from 21.5 percent in wheat, 30.3% in rice, 22.6% in maize, 17.2% in potato, and 21.4% in soybean [4]. Therefore, it is crucial to develop effective weed and disease management techniques to minimize yield losses and reduce chemical residues on crops to ensure global food security. The introduction of precision agriculture, i.e. applying modern technology, such as Unmanned Aerial Systems (UAS) and unmanned ground vehicles (UGV), to farming practices, could help increase quantity and quality of produce while improving the sustainability of the food supply [5].

In agriculture, UASs are commonly used as a remote sensing platform to assess and monitor crops. Emerging applications include precision distribution of chemicals, biological control agents, and livestock health monitoring [6]. Significant developments in Global Positioning System (GPS), Machine Vision (MV), and laser-bases systems have enabled UGVs to detect maturity of fruits, harvest crops, and spray herbicides and pesticides on affected areas [7].

Simulators play a critical role in robotics research as tools for quick and efficient testing of new concepts, designs, and algorithms. Gazebo is an open-source robotics simulation platform capable of creating 3-D dynamic multi-robot environments with a wide range of use cases. It readily integrates with Robot Operating System (ROS) and software written for simulated robots can be deployed to real robots with minimal changes to code [8]. In the case of UASs, the ROS/Gazebo architecture has been extended to simulate multiple co-operative UASs [9]. Furthermore, many open-source flight controllers and their flight control software such as Pixhawk and PX4 readily integrate with ROS and Gazebo [10].

Advancements in the fields of machine learning (ML) and robotics have led to the development of more adaptable vehicles in agriculture. Recently, Deep learning (DL) has gained momentum in the agricultural domain. It has been tested in several agricultural applications including identification of weeds, land cover classification, plant recognition, counting fruits, and crop type classification [11]. Pre-trained models for image classification and algorithms such as YOLOv3 [12] for object detection have been effective in identifying single and multiple weeds, given sufficient data [13].

Other authors have implemented vision-based multi-robot simulations, such as in [14]. However, there are no SITL packages that incorporate deep learning models for target detection while establishing a direct line of communication

between UASs and UGVs in agriculture. In this study, we bridge this gap by providing a comprehensive SITL package to test software and algorithms in co-operative UAS-UGV system for use in precision agriculture. The UAS provides an aerial snapshot of the field and identifies areas of interest, such as weeds or diseases, using the YOLO real-time object detection system. The location co-ordinates of the target is then communicated to the UGV. The UGV, equipped with weed and disease management tools, is then be deployed to the location of interest.

II. Methods

A. UAS Design and PX4 Integration

PX4 [15] is an open-source autopilot software. It can be embedded with open-source flight controllers such as Pixhawk for vehicle control. It contains multiple packages and libraries for integration of multi-rotors, fixed-wing, VTOLs, and rovers. PX4 is also available for SITL in the Gazebo simulation platform, jMAVSim, and others. A DJI Matrice 600 model was designed for integrated with the PX4 firmware. An overview of the complete file structure is provided in Fig. 1.

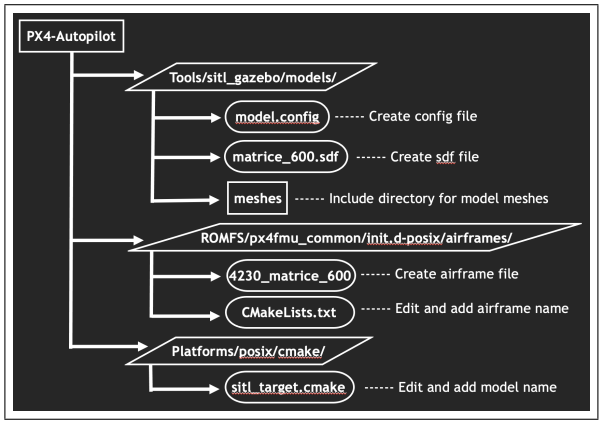

Fig. 1. File structure within the PX4 firmware to create a custom model for SITL in Gazebo.

The model is defined within the PX4-Autopilot firmware directory. Under the models folder of Tools/sitl_gazebo, model.config and model.sdf files are created. An STL model of the DJI M600 is placed within the meshes folder. The model is then defined in the cmake file sitl_target, which enables the model to be used in the Gazebo simulator. An airframe file is created which defines the type of UAS, in this case a hexacopter. Further, the Proportional-Integral-Control (PID) gains are adjusted here. This airframe file is then defined in CMakeLists. The airframe file can be adapted accordingly for defining the UGV.

The model (Fig. 2(a)) is primarily designed in the SDF file for use in Gazebo. Various plugins available with the package are used to define the propeller motors, camera, camera gimbal, Inertial Measurement Unit (IMU) sensor, and Global Positioning System (GPS) module. In order to ensure stable flight, it's imperative that the inertia tensors are accurate. For simplicity, the inertia tensor for the main

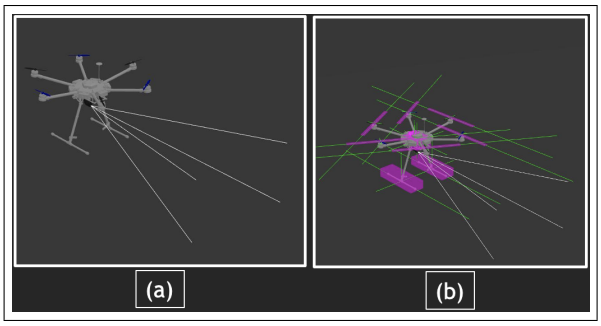

Fig. 2. (a) DJI M600 model. (b) Model with inertia overlay.

body was obtained by modeling the chassis and landing legs as cuboids (Fig. 2(b)). The propellers were modeled as extremely thin cuboids. The inertia tensors for the cuboids were determined as follows, where w is width, h is height, d is depth, and m is mass:

$$I = \frac{1}{12}m \begin{bmatrix} (h^2+d^2) & 0 & 0 \\ 0 & (w^2+h^2) & 0 \\ 0 & 0 & (w^2+d^2) \end{bmatrix}$$

Several plugins are used to simulate camera features in the Gazebo world. The GStreamer plugin libgazebo_gst_camera_plugin.so is set to stream on UDP port 5600 which enables live video feed from the Gazebo world to QGroundControl. The gazebo_ros_pkgs camera plugin libgazebo_ros_camera.so will publish to a set topic that can later be used by the YOLO ROS2 node for object detection. In this case, the topic is set to camera/raw_image.

B. PX4-ROS2 Bridge

In a physical UAS system, the flight controller and flight computer are delegated to separate modules. The controller is responsible for various aspects of flight such as stabilization and navigation. The computer is responsible for object detection, UAS-UGV communication, and mission planning. The flight computer tasks are run as ROS2 nodes, which can exchange information with PX4 uORB topics through the PX4-ROS2 Bridge using the microRTPS protocol.

The microRTPS Bridge consists of a client running on PX4 and an agent running on the ROS2 computer. The agent side of the Bridge is managed by the px4_ros_com package which can be sourced from the PX4 GitHub. In the newer PX4 distributions, the client is started automatically. Once simulation has been launched, the microRTPS agent is started and the UDP ports for the PX4 vehicle instances is matched to establish connection.

Despite its name, ROS is not an operating system like macOS or Windows, but rather a software framework for robotics development. It is an open-source, peer-to-peer, tools-based, and multi-lingual framework that allows for the development of robot software systems applicable to a variety of hardware platforms, research settings, and runtime requirements [16]. In this study, ROS2 Foxy was used to take advantage of the DDS/RTPS security features, multi-platform capabilities, and extended support offered over its predeces-

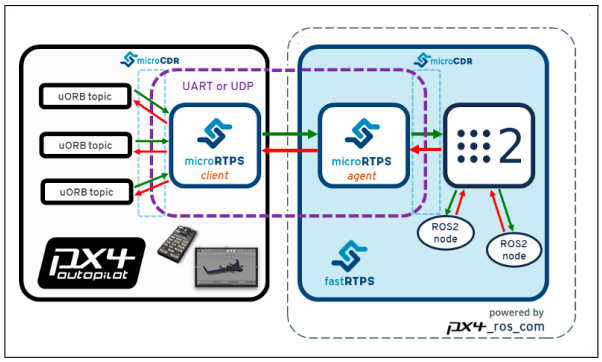

Fig. 3. PX4-ROS2 Bridge. This enables exchange between PX4 uORB topics and ROS2 nodes. [15]

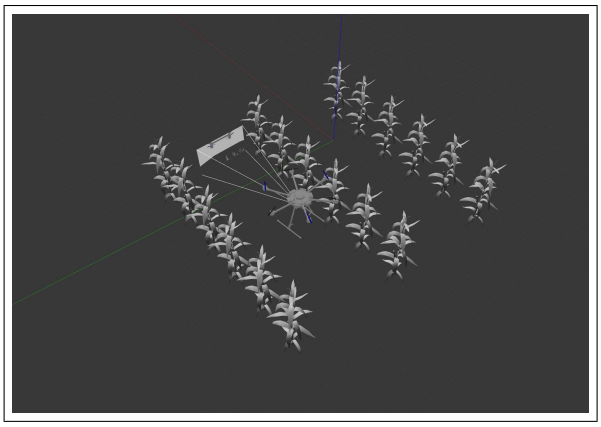

Fig. 4. DJI M600 PX4-based SITL in simulated Gazebo farm world.

sor ROS1. However, since ROS2 is fairly new, multi-vehicle cooperation is not well studied using ROS2. The advantage of this setup is that software developed for SITL testing can be deployed to physical system with minimal changes to code. The structure of the bridge is provided in Fig. 3.

C. YOLO ROS

The darknet_ros package [17], which integrates YOLO as a ROS2 node, was used to implement object detection. Camera images from the Gazebo simulator are published via the PX4-ROS2 bridge to The YOLO ROS2 node. In order to use YOLO with ROS2, certain parameters must be adapted in the darknet_ros package. In the darknet_ros/config/ros.yaml, the image subscriber must subscribe to the topic published by the Gazebo simulated camera. In this case, the topic is camera/image_raw. Similarly, these parameters must also be changed in the launch file. Pre-trained weights and config files must be added to the corresponding folders.

D. UAS-UGV Communication

The UAS flies a set flight plan that can be set via the ground station software QGroundControl. The UAS communicates location co-ordinates of the target identified by YOLO to the UGV using the Micro Air Vehicle Link (MAVLink) [18] communication protocol. The UGV then follows the flight plan to the target for further inspection and treatment.

III. Results and Discussion

The PX4-based Gazebo SITL successfully simulated a stable UAS (Fig. 4) with way-points programmed using QGroundControl. This indicates satisfactory modeling of the moment of inertia tensors in the model SDF files, airframe PID tuning, and PX4 integration.

Using the default PX4 Gazebo SITL environment, limited models can be spawned for multi-vehicle simulation using the bash script gazebo_sitl_multiple_run.sh. This sets up multiple MAVLink UDP ports for QGroundControl and microRTPS client. For ROS2 integration, microRTPS agents must be started for each PX4 MAVLink instance. The primary issue with this method of multi-vehicle simulation is that there's no direct line of communication between the vehicles. This is because only two lines of communication are opened in each vehicle instance. The first is the MAVLink connection between the PX4 instance and QGroundControl and the second is the microRTPS link between the PX4 client the ROS2 agent. Therefore, a separate link must be established between the ROS2 computers for each vehicle. Preliminary implementation with the ROS wrapper for MAVLink known as MAVROS have been unsuccessful due to MAVROS's current instability with ROS2. This is because MAVROS was optimizied for use with ROS1.

The darknet_ros package was set up using pre-trained weights from Pascal VOC and Microsoft COCO data sets. Custom detection objects can be enabled by providing weights and configuration files on the weights and cfg directories in the package. In the Gazebo simulation environment, simulated VOC and COCO objects like truck, car, person, stop sign, and so on were detected successfully (Fig. 5). Preliminary tests were performed on CPU with an average of 1.2 FPS. However, using an NVIDIA GPU can enable the parallel computing capabilities of Compute Unified Device Architecture (CUDA) making it about 500 times faster.

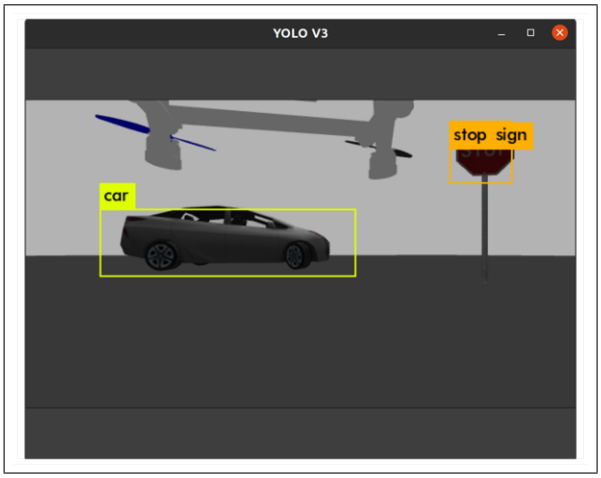

Fig. 5. Preliminary results of YOLOv3 object detection in Gazebo simulation using VOC and COCO pre-trained weights.

IV. Future Work

Acknowledgements

This project was supported by the generosity of the Engineering Undergraduate Research Office (EURO), the Department of Agricultural and Biological Engineering, and the Digital Agriculture Discovery (DAD) Lab through the Summer Undergraduate Research Fellowship (SURF) at Purdue University.

References

[1] "2022 global report on food crises," Global Network Against Food Crises, 2022.

[2] D. o. E. United Nations and P. D. Social Affairs, "World population prospects 2022: Methodology of the united nations population estimates and projections (un desa/pop/2022/tr/no. 4)," 2022.

[3] J. Gao, D. Nuyttens, P. Lootens, Y. He, and J. G. Pieters, "Recognising weeds in a maize crop using a random forest machine-learning algorithm and near-infrared snapshot mosaic hyperspectral imagery," *Biosystems Engineering*, vol. 170, pp. 39–50, 6 2018.

[4] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," pp. 436–444, 5 2015.

[5] R. Gebbers and V. I. Adamchuk, "Precision agriculture andfood security," pp. 825–828, 2 2010.

[6] D. van der Merwe, D. R. Burchfield, T. D. Witt, K. P. Price, and A. Sharda, "Chapter one - drones in agriculture," ser. Advances in Agronomy, D. L. Sparks, Ed. Academic Press, 2020, vol. 162, pp. 1–30. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065211320300328

[7] R. Rahmadian and M. Widyartono, "Autonomous robotic in agriculture: A review." Institute of Electrical and Electronics Engineers Inc., 10 2020.

[8] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," 2004. [Online]. Available: http://playerstage.sourceforge.net/gazebo/

[9] C. Bernardeschi, A. Fagiolini, M. Palmieri, G. Scrima, and F. Sofia, "Ros/gazebo based simulation of co-operative uavs," J. Mazal, Ed., vol. 11472. Springer International Publishing, 2018. [Online]. Available: http://link.springer.com/10.1007/978-3-030-14984-0

[10] E. Ebeid, M. Skriver, K. H. Terkildsen, K. Jensen, and U. P. Schultz, "A survey of open-source uav flight controllers and flight simulators," *Microprocessors and Microsystems*, vol. 61, pp. 11–20, 9 2018.

[11] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," pp. 70–90, 4 2018.

[12] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[13] A. Ahmad, D. Saraswat, V. Aggarwal, A. Etienne, and B. Hancock, "Performance of deep learning models for classifying and detecting common weeds in corn and soybean production systems," *Computers and Electronics in Agriculture*, vol. 184, 5 2021.

[14] K. D. Nguyen and T.-T. Nguyen, "Vision-based software-in-the-loop-simulation for unmanned aerial vehicles using gazebo and px4 open source," 2019.

[15] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6235–6240.

[16] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," 2009. [Online]. Available: http://stair.stanford.edu

[17] M. Bjelonic, "YOLO ROS: Real-time object detection for ROS," https://github.com/leggedrobotics/darknet$_ros$, 2016 − −2018.

[18] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, "Micro air vehicle link (mavlink) in a nutshell: A survey," *IEEE Access*, vol. 7, pp. 87 658–87 680, 2019.