AGR-o-RAMA  
Agricoltura di precisione con Robot Autonomi per il Monitoraggio Attivo

Deliverable D2.1  
Modulo di simulazione foto-realistico 3D per droni in agricoltura  
WP2

**Informazioni sul documento**

| **Codice Progetto** | |  | | **Acronimo** | | AGR-o-RAMA | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Titolo** | | | | Agricoltura di precisione con Robot Autonomi per il Monitoraggio Attivo | | | |
| **Bando** | | | |  | | | |
| **Data di avvio** | | 15/04/2021 | | **Durata** | | 24 mesi | |
| **URL di progetto** | | | | www.agrorama.it | | | |
| **Responsabile di Progetto** | | | | Vito TRIANNI | | | |
| **Deliverable** | | | | D2.1 | | | |
| **Work Package** | | | | WP2 | | | |
| **Data consegna** | **Contrattuale** | | M9 | | **Attuale** | | M9 |
| **Natura** | | Report | | **Disseminazione** | | Confidenziale | |
| **Beneficiario Responsabile** | | | | Sapienza | | | |
| **Autore principale** | | | | Carlos Carbone | | | |
| **Co-autore(i)/Revisore(i)** | | | | Si veda tabella seguente | | | |

**Cronologia delle revisioni**

| Versione | Data rilascio | Satus | Cambiamenti | Autore |
| --- | --- | --- | --- | --- |
| 0.1 | 10/01/2022 | Bozza |  | Vito Trianni |
| 1 | 15/01/2022 | Finale |  | Carlos Carbone |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

## 

## Introduction

Simulation is a fundamental tool for swarm robotics research. However, most simulators are developed from scratch for specific robot or mission conditions, thus commonly validating swarm strategies in highly controlled environments.

Highly realistic simulators are commonly not mandatory in research to validate swarm strategy theories. However, simplified or abstract simulators suffer from a large gap between the simulated and real world behavior which may set back the transition from the simulator to real world implementations. The magnitude of this problem is much more prominent in the case of Aerial Swarm Robotics where deployment of real hardware is complex and errors can have strong consequences in terms of hardware damage and money losses. Furthermore, the precision agriculture context adds additional difficulties since the state of the crop changes over time and experiments commonly require large outdoor spaces. Therefore, highly realistic simulation environments are much more essential to aerial swarm robotics research in precision agriculture. Moreover, the possibility to address complex vision problems such as classifying crops and weeds is usually not available in simulators, especially for robot swarms. These simulations bring other challenges since photo-realistic rendering in real time is a heavy computational task which also scales by adding multiple robots with cameras.

## Description of the simulator

**Flightmare**

Flightmare provides a framework that uses the Unity rendering engine capabilities to build a photorealisitc 3D simulation environment, while the *UAV* dynamics are simulated separately with parallel programming. Both simulations are then coordinated with the asynchronous messaging library ZeroMQ providing a high level of modularity to switch simulation components, including the specific simulated quadrotor dynamic model.

Flightmare includes three modular quadrotor *UAV* dynamic simulation solutions which serve different purposes. The Gazebo based dynamics solution is slower compared to the rest but it provides higher fidelity physics simulation based on its physics engine Bullet. Furthermore, Gazebo provides an environment compatible with *ROS* which is an essential tool for real world experiment applications as many real robots rely on it for real world deployment. However, the Flightmare setup aims to increase the compatibility of Unity with multiple robotics tools rather than focusing on exploiting possible simulation conditions within Gazebo and *ROS* where simulated *UAV* count is limited. Flightmare can support hundreds of simulated quadrotor *UAV* agents inside Unity. However, this takes place without Gazebo and using a less accurate dynamic solution to provide a high number of state transition data for deep reinforcement learning purposes.

Unity within Flightmare provides simulation of *UAV* cameras, ground-truth depth, semantic segmentation and range finders. Thus, it allows the user to switch between these and add them or remove them offline or online. Furthermore, point cloud environments can be extracted and exported for path planning development. Additionally, the Flightmare paper states that the developed simulation environment could provide very strong support to *SLAM* research with a high number of 3D environments available in the Unity Asset Store. Other, demonstrations include virtual reality implementation for safe human-robot interaction

The Flightmare simulator demonstrated capabilities compose a good fit for our needs in this research. The Unity/Gazebo setup serves as a starting point that provides the main tools required for realistic simulation of a *UAV*: Highly realistic *UAV* dynamics and a photo-realistic rendering engine.

The Unity/Gazebo default setup in Flightmare can spawn one *UAV* in the simulation environment which is present in both Unity and Gazebo simultaneously. Coordination of data sharing for imagery and *UAV* pose is managed through *ROS* which is a middleware that serves for information sharing between the simulation software. Thus, Unity’s *UAV* pose is replicated through *ROS* with the pose data of the *UAV* from the Gazebo simulator, likewise camera feed from Unity is provided back to *ROS*. Furthermore, the *ROS* implementation allows us to add further modular simulation features required. This feature is also key for implementing later a *CNN* for online plant detection.

The roles of the simulation software within our simulation setup are summarized for clarity purposes as follows:

• *ROS*: Middleware used for data sharing between each simulation component.

• High level swarm control system.

• Image classification system.

• Unity: Photo-realistic rendering engine for plant perception simulation.

• Gazebo: High fidelity physics simulation environment for simulation of *UAV* dynamics.

Our high level swarm control system is implemented within *ROS* to develop the proposed swarm strategy behavior, see Section 5.5. Additionally, an image classification system based on a *CNN* is implemented and coordinated with the control system through *ROS*, see Section 5.4.1.

The next steps in the simulation development are: to implement the weed fields within the Unity setup in Flightmare and to implement the *UAV* swarm deployment within Gazebo and *ROS* to achieve the realism required for simulation of multiple *UAVs* in precision agriculture.

The crops are spawned procedurally in Unity within Flightmare with a similar approach to the one presented in Chapter 3. Leaves are spawned using plant structure scripts to spawn plants which are then distributed around the field. The fields in Chapter 3 with row/column field shape for the growing plant and randomized distributed weed placement serve well to generate *CNN* training imagery.

### **Precision agriculture with Flightmare**

The Unity setup within Flightmare includes generic environments obtained from the Unity Asset Store. These serve mainly for demonstration purposes in the Flightmare general setup which aims to provide a simulator where other 3D environments could be added for further research. The setup includes additional scripts for Unity scene and object management required during simulation which can be included in other implementations if needed.

The inclusion of a simulated crop field is required for the precision agriculture experimental setup. Flightmare provides a Unity project to build executable files to be included in the Flightmare Gazebo setup. The fields from Chapter 3 are then added into this project to be included into the simulator. Figure 1 shows the Flightmare simulator with a sugar beet field inside Unity while Gazebo handles the quadrotor *UAV* dynamic simulation. Figure 2 shows the camera view from one of the *UAV* agents.

The crops are spawned procedurally in Unity within Flightmare with a similar approach to the one presented in Chapter 3. Leaves are spawned using plant structure scripts to spawn plants which are then distributed around the field. The fields in Chapter 3 with row/column field shape for the growing plant and randomized distributed weed placement serve well to generate *CNN* training imagery. However, the weed distribution in this new setup is procedurally spawned based on cells with the distribution methodology approach used in Chapter 4. Now we limit the possible amount of weeds up to 9 per cell, where the cell is subdivided into 3x3 portions where each weed can spawn from a random position.

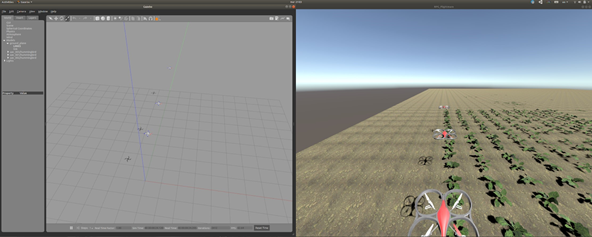


Figure 1: Screenshot sample of the Flightmare simulator with the sugar beet field from Chapter 3. Gazebo (left) and Unity (right)

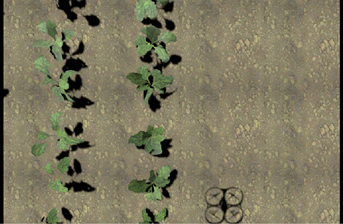


Figure 2: Screenshot of camera view from one of the *UAV* agents from Figure 1.

### **Swarm simulations with Flightmare**

After reviewing the simulator in depth, we noticed that additional code is required to properly enable Flightmare to spawn multiple coordinated *UAVs* by using the Gazebo/Unity simulation setup where the simulation coordination takes place in *ROS*. The Flightmare’s scripts and *GUI* provide an easy entry point but mainly for the setups where Unity is the main interface for the user to interact with, which is not the case when Gazebo is included. In the Gazebo/Unity setup, the simulation only includes one *UAV* and the default Flightmare *GUI* in Unity seems to serve only to spawn different 3D environments. Therefore, additional code is required in Flightmare with the Gazebo setup to simulate a swarm of quadrotor *UAVs*. It is worth mentioning that the amount of *UAVs*

that can be simulated simultaneously in Gazebo is much more limited because of its simulation fidelity compared to the other quadrotor *UAV* dynamic solutions in Flightmare.

Multiple *UAV* agents need to be spawned simultaneously in the field in random positions within the field and each agent is required to have a camera attached in Unity for visual feedback and have their corresponding pose replicated from the Gazebo simulation.

The *ROS* structure is based on "packages" containing source code to be executed during simulation in which each software instance is called a "node". Flightmare [2] includes a set of *ROS* packages providing a starting point that ensures data sharing between Unity and *ROS*. This is highly useful since Unity is a stand alone program that cannot be run as a *ROS* node thus it requires additional code to interact with *ROS*. Thus, the main Flightmare’s *ROS* packages that required additional code to deploy a *UAV* swarm are the following:

• Flightrender package: Includes the code that spawns Unity within the *ROS* environment.

• Flightros package: Includes the code that coordinates data exchange between Unity and Gazebo.

Additional code is added to Flightros to ensure that multiple *UAVs* can be simulated simultaneously in Gazebo with replicated *UAV* poses shared to Unity. Furthermore, Flightrender includes the executable version of the Unity project for the simulated field which needs to spawn the weeds in the corresponding cell locations.

*ROS* provides a controlled messaging interface between nodes to ensure reliable and robust parallel program execution and coordination for robot simulations. Moreover, it support these features over network with multiple computers meaning multiple real robots, which is required for real world applications.

The rest of Flightmare packages and Flightmare dependency packages did not required further intervention to properly work with the developed simulation environment. Thus at this point, Flightmare is ready to communicate with the developed packages where a control node can take decisions based on the simulator data feedback. This swarm control node is spawned per each *UAV* and it is built upon the code developed for Chapter 4 which is modified to be properly implemented in *ROS* and in real *UAVs* as well.

## Conclusions

The developed simulation environment includes a high number of realistic features usually neglected or not simulated in swarm robotics. However, it is also clear that the considered conditions can be further studied and measured to ensure that the simulation environment truly covers all the difficult and complex real limitations that can be present during the deployment of real *UAVs*. This is a difficult task since hardware is rapidly improving as well. However, this still represents a research area worth studying considering how realism is such a prominent need in aerial swarm robotics which is also growing as a research area. Furthermore, the modularity of the simulation environment enables the user to include additional features that may be updated as technology moves forward. Sensors that may be included in *UAVs* for precision agriculture practices can be simulated and included into the simulation framework. For example, the inclusion of the *NIR* sensor is achieved by a simple approach of synchronizing *RGB* data gathering with a sensor that could not be modeled within Unity. Likewise, additional imagery sensors could be included in this way for new types of technologies.

1. J. Snape, J. v. d. Berg, S. J. Guy, and D. Manocha. The Hybrid Reciprocal Velocity Obstacle. IEEE Transactions on Robotics, 27(4):696–706, Aug. 2011.