# V-REP Integrated Paparazzi Simulation

Dominik Weikert

***Abstract*—This paper describes a framework for integrating the native Paparazzi control-loop for UAVs into the 3D virtual-robot experimentation platform V-REP, created for the simulation of the quadcopters used in the Otto-Von-Guericke University SwarmLab.**

## I. Introduction

The OVGU SwarmLab is a robotic lab in which experiments applying the theory of swarm intelligence are performed, mainly using custom-built quadcopters called FINken. To complement the live experiments, a 3-D simulation software is needed. Simulation allows for faster creation of experimental data and reduces the risk of severely damaging the copters in the lab. Additionally, it enables the exploration of a wide range of environments and conditions not reproducible in a closed lab.
The real FINken in the SwarmLab use paparazzi, an open-source software for the control of multicopters and other UAVs. As Paparazzi already provides a Simulator, the New Paparazzi Simulator (NPS), which allows custom back-ends to provide a flight dynamics model (FDM), it seemed the best do create such a back-end to integrate into the customizable virtual robot experimentation platform V-REP [1]. This provides the advantage of implicit portability between the real FINken running Paparazzi and the V-REP - Paparazzi simulation. To this end, a custom V-REP plugin was developed in C++ to enable the exchange of data between the two programs with the goal to provide a swarm-capable, easily extendible, fully 3-D physics enabled and scalable simulator for quadcopters. The V-REP plugin communicates with the Paparazzi FDM and provides it with the necessary data regarding the copter attitude obtained in the V-REP simulation. Paparazzi can then use the data provided in the FDM in the control loop for the copter as it would for a real aircraft and calculate the commands to be sent to the motors, which can then be used to continue the simulation in V-REP, completing the cycle. As such, Paparazzi is still responsible for the complete control of the quadcopters while V-REP provides the physics simulation, attitude data and visualization. This paper describes the basic architecture of the framework created to establish this functionality, and provides some evaluation data on the accuracy of the V-REP simulation loop.

### A. Related work

Other simulation software already exists for the simulation of robots. The most widely used simulators are shortly described in this section, and reasoning as to why they are not suitable for use in the SwarmLab will be given.
JSBSim is an open source Flight Dynamics Model written in C++ [2]. JSBSim functionality is already provided as the base simulator used in the Paparazzi software, as such it could easily be used to simulate quadcopter models specified by Paparazzi without the need to rewrite code. However, JSBSim does not provide a swarm capable simulation framework and as such was not suitable for the swarm-focused work done in the SwarmLab.
Another widely used simulation software is Gazebo [3], which provides a 3-D physics engine and interactive scenes during simulation and is extensible using C++ plugins. However, the physics engine provides no particle systems and only a limited GUI is available, e.g. all scene editing is done using xml files.

ARGoS provides many of the features that were sought after in the SwarmLab: It is modular and capable of parallel simulation of swarms of robots with high scalability [4]. However, its simulation is limited to 2-D environments and most SwarmLab experiments take place in 3-D.
V-REP was chosen as a physics simulator as it provides all the features needed while including an intuitive GUI to easily create complex scenes and manipulate the simulation.

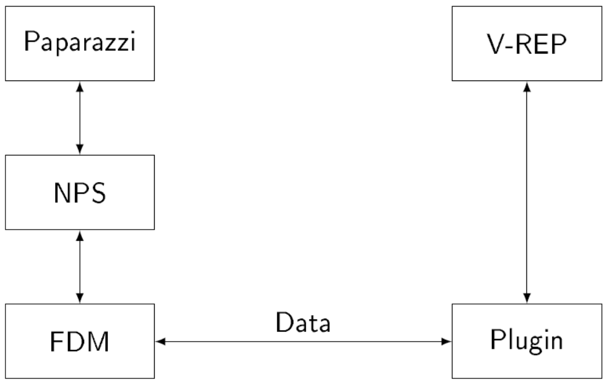

Fig. 1: Basic architecture model

## II. Framework Architecture

The base of the combined Simulator is founded on a client-server architecture, with the Paparazzi controller acting as a client to the V-REP plugin A basic outline of this architecture is illustrated in Figure 1: In the FDM of the Paparazzi simulator, a connection to a server representing the controlled simulated copter in V-REP is established. Using a multi-threaded model, multiple copters could be simulated at any point in time. Once the connection is established, simple data packets are exchanged containing the necessary data for each side:

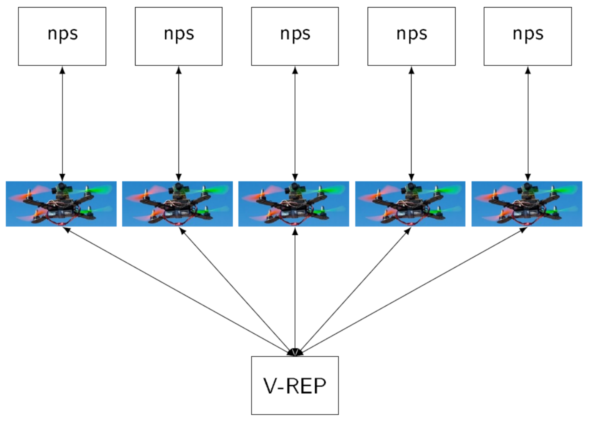

Fig. 2: Architecture using multiple simultaneous connection

Paparazzi provides data regarding the commands for the control of the FINken-motors, while V-REP provides data regarding the position and attitude of the FINken needed in the FDM to compute those controls.

## A. V-REP plugin

The V-Rep plugin is responsible for providing the server architecture as well as handling the data on any individual FINken and synchronization between them. To this end, each incoming client connection is paired (via ID) with a compatible FINken in the V-REP scene. This FINken object can then run in a separate thread to obtain its commands, update its position and send its new position to the Paparazzi client. Meanwhile the plugin ensures that all FINken have received (or sent) their data before the simulation can continue. The basic loop in the plugin thus consists of the following steps:

1: Receive commands from the Paparazzi client
2: Apply commands to the copters
3: Wait for all copters to update their position
4: Send new positions to the paparazzi client

## B. Paparazzi back-end

As described above, the Paparazzi client is responsible for receiving the copter data, passing it to the Paparazzi simulator and sending the new commands to the V-REP plugin.

When the position and attitude data is received, it is converted from the V-REP coordinates, which are in the NED system, to the correct coordinate systems used in Paparazzi, mainly ENU and ECEF. Paparazzi already provides several data structures and converters to accomplish this. Once those data structures are filled correctly, the Paparazzi simulation can advance and calculate the next commands to send to the V-REP plugin. Afterwards, the client blocks the advancement of the NPS until new attitude data is received. To keep the simulations in time-sync, Paparazzi also receives the time-step set in the V-REP simulation and the simulation time is manually advanced using that time-step.

A complete overview of the full simulation loop including data exchange can be seen in Figure 3.

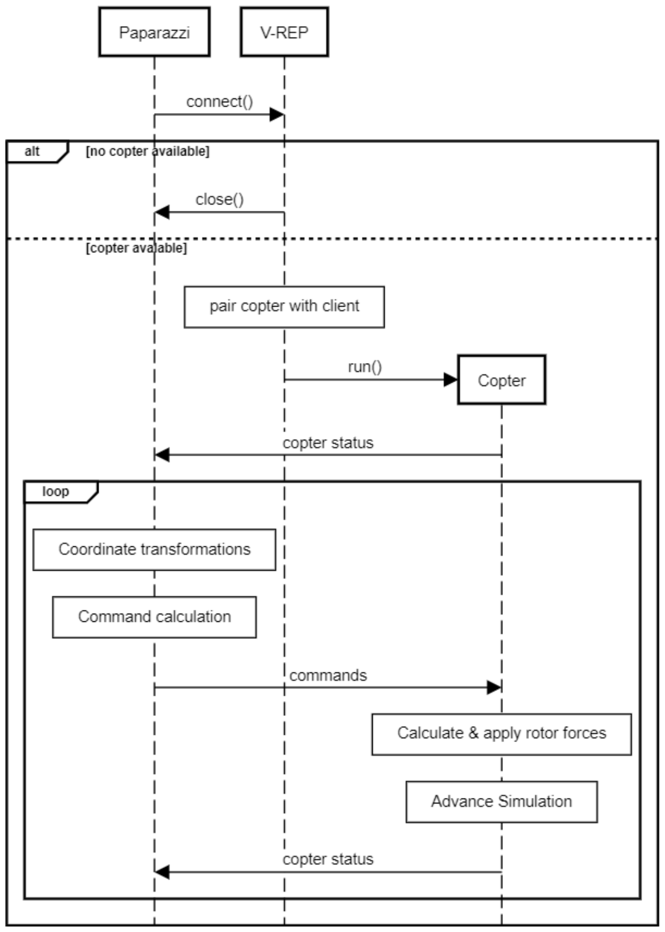

Fig. 3: Basic simulation loop sequence

## III. Evaluation

Several extended test flights were conducted to verify the accuracy of the simulated control loop. This section will detail some of the results of these test flights.

The the first test flight is on a horizontal line between points 10m apart. The results are shown in Fig 5. The figure shows the stability of the copter and the correct control dependency between the roll command and the acceleration in the y direction, denoted as y_d.

Figure 6 shows the positioning for a square shaped test flight around the origin of the local coordinate system, namely the points $(5,0), (0,5), (-5,0)$ and $(0,-5)$. This shows the exact positioning throughout the flight with the copter reaching maximum position in one coordinate when the other coordinate is 0.

## IV. Future work

The framework described in this paper is still a work in progress. While the basic simulations loop is complete and correctly functioning for a single copter, the synchronization between multiple copters still needs to be implemented. Another planned feature is the correct implementation of sensor noise (including the implementation of a wide range of sensors usually used on quadcopters), with the parameters for that noise directly specifiable from the simulation interface.

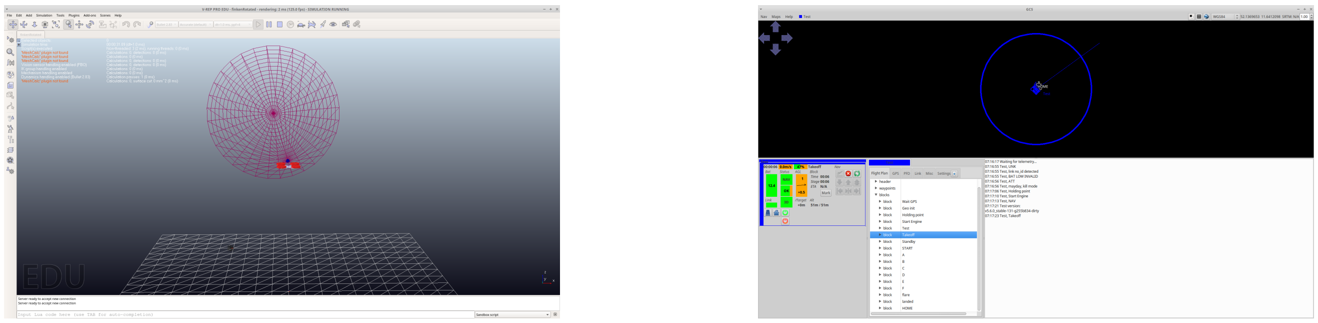

Fig. 4: Screenshots of both simulation parts: V-REP user interface with simplified copter model (left) and Paparazzi control center(right)

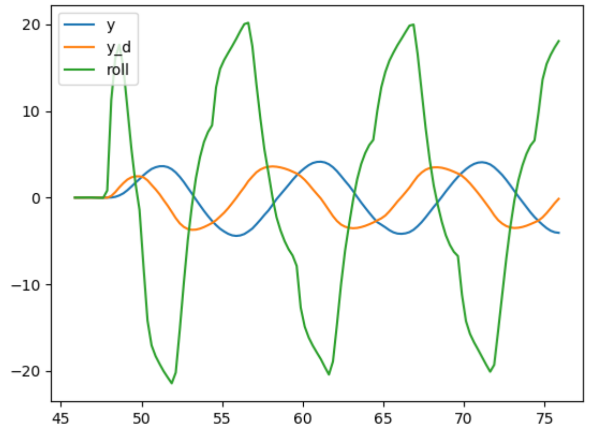

Fig. 5: Test-flight 1: straight line on x axis

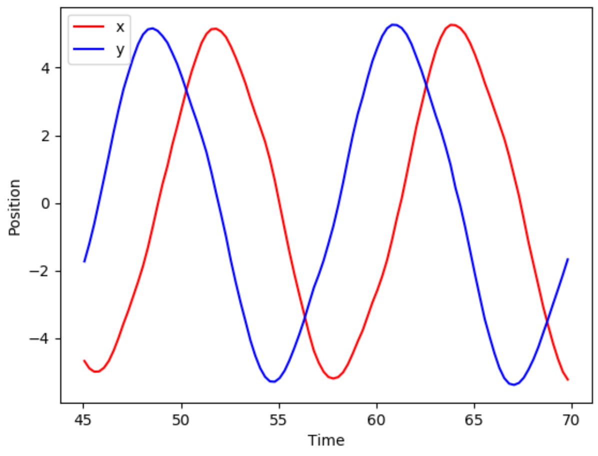

Fig. 6: Test-flight 2: square around origin of local coordinate system

Additionally, some environmental features such as wind will be implemented using a particle physics system.

## References

[1] E. Rohmer, S. P. Singh, and M. Freese, “V-rep: A versatile and scalable robot simulation framework,” in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 1321–1326.

[2] J. Berndt, “Jsbsim: An open source flight dynamics model in c++,” in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, 2004, p. 4923.

[3] N. P. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator.” Citeseer, 200.

[4] C. Pinciroli, V. Trianni, R. OGrady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle *et al.*, “Argos: a modular, parallel, multi-engine simulator for multi-robot systems,” *Swarm intelligence*, vol. 6, no. 4, pp. 271–295, 2012.