Realistic drone model in RotorS for indoor navigation

Jorge García Pueyo (758807), Juan Dendarieta (538123)

***Abstract*— Real world testing of Micro Aerial Vehicles has a wide range of disadvantages like the need of access to expensive hardware or difficulty to reproduce results and debug errors. Simulation suites, like Gazebo, exists to reduce these disadvantages. We have developed a simulation model of a real hexarotor available at RoPeRT - Universidad de Zaragoza inside RotorS, a framework to simulate Micro Aerial Vehicles on Gazebo. The model was evaluated in a simulation by performing a trajectory navigation task in the inside of a simulated house. This project lays the foundations for the development of new algorithms for the hexarotor and its quick testing on simulation software that accurately matches reality.**

I. Introduction

Testing of new algorithms on Micro Aerial Vehicles (MAVs) in the real world requires access to expensive hardware, substantial amount of time to setup and possible errors are hard to reproduce with a chance of damaging the hardware. Simulation suites, like Gazebo [1], are developed to reduce field testing and the associated risks. They are also useful for situations where access to the real hardware can not be granted, for instance to students. However, simulation requires to reproduce reality faithfully, if not, problems might arise when porting the newly developed algorithms to the real world.

Indoor environments pose challenges for navigation of autonomous robots like collision-avoidance without GPS-data or small spaces for maneuvering. Traditionally, exploration of such spaces was done by ground robots but there has been a growing interest in using airborne vehicles flying autonomously in indoor spaces, especially in those where the conditions make the usage of land based robots impossible. Underground tunnels are one of such challenging indoor environments for ground robots because corridors have usually irregular floors that might even be flooded. There are also spaces that might be inaccessible for ground vehicles, such as vertical vertical corridors, and it is in these situations where we could benefit from using MAVs.

In this project, we have:

- Modeled a 3D realistic model of Tyrion, an hexarotor available at RoPeRT - Universidad de Zaragoza.
- Integrated Tyrion in RotorS [2], a MAV Gazebo simulator, defining the real physical properties and sensors.
- Analyzed the navigation of Tyrion, including its sensors, in an indoor house scene.

II. Background

In this section, we present the theoretical background about how MAVs are controlled to flight and navigate 3D space.

A. MAV modeling

The hexarotor is a MAV that consists of six motors on which propellers are fixed and attached at the ends of six arms in a symmetric composition (See figure ...). It is an under-actuated system where the desired output trajectory can only be defined by a 4-dimensional output, namely the 3d position of the center of mass and the yaw, in spite of the presence of six control inputs (even though the hexarotor has six propellers the dynamics are similar to the quadrotor [3] [4] [5]).

1) Single rotor force: The steady-state thrust generated by a rotor at velocity ω_i is defined by $T_i = c_T w_i^2$ where c_T is the rotor thrust constant.

2) Dynamics: The dynamics of the hexarotor are well known [6]. The total thrust applied to the frame is the sum of the thrusts from each individual rotor. The total net moment arising from the aerodynamics $\tau = (\tau_1, \tau_2, \tau_3)$ is a combination of the rotor forces and the air resistance. We can write this in matrix form:

$$\begin{bmatrix} T_\Sigma \\ \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix} = \begin{bmatrix} c_T & c_T & c_T & c_T \\ 0 & dc_T & 0 & -dc_T \\ -dc_T & 0 & dc_T & 0 \\ -c_Q & c_Q & -c_Q & c_Q \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \quad (1)$$

where c_Q is the rotor moment constant. The equations of motions of the MAV are

$$m\dot{v} = -mga_3 + T_\Sigma R \quad (2)$$

$$J\dot{\Omega} + \Omega \times J\Omega = \tau \quad (3)$$

where v is the linear velocity in the inertial frame, g is the gravity force, R is the rotation matrix from the body-fixed frame to the inertial frame, Ω is the angular velocity in the body-fixed frame and J is the inertial matrix with respect to the body-fixed frame.

3) Control: As seen in Eq. 1, only the total thrust and moments can be controlled directly. Therefore, to navigate in 3D space, the vehicle has to be tilted towards the goal point. This is referred as attitude controller and its dynamics are usually much faster than the translational dynamics and it is the reason why often a cascaded control approach is used. For this project, a geometric tracking control [7] is used.

4) Estimating state: Cascading controllers are separated in a position loop and attitude loop. The outer position loop needs the position p and linear velocity v while the inner attitude loop needs the orientation $R = (\phi, \theta, \psi)$ and the angular velocity Ω. To estimate this values, MAVs have sensors like an Inertial Measurement Unit (IMU) for the orientation of GPS for the position. In indoor scenarios, GPS-data is not precise enough so marker-based localization like

VICON or visual-odometry or visual SLAM is used through different types of cameras and LIDAR.

B. RotorS

RotorS [2] is a MAV simulation framewok built on top of Gazebo [1] and the Robot Operating System (ROS) [8]. The objective is to reduce testing times, separate problems for testing (like a hardware failure in a real scenario from algorithm bug in a new local planner) making debugging easier and, ultimately, reduce the number of crashes of real MAVs.

An overview of the main components of RotorS is shown in Fig. 1, where the left side corresponds to the components of a simulated MAV in RotorS and the right part the components of a real MAV. Ideally, both parts are analogous such that the migration to reality is done without changes.

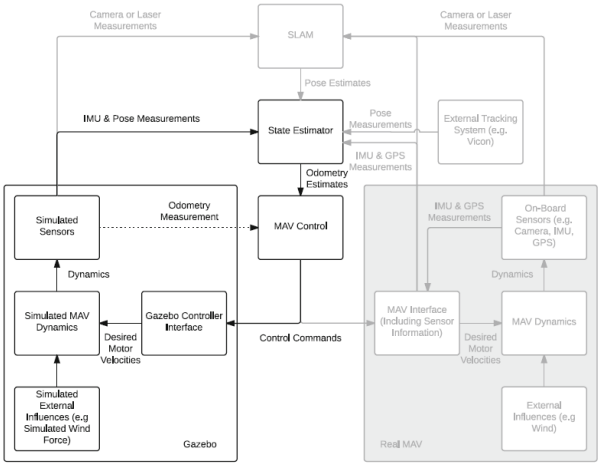


Fig. 1: Building Blocks for a MAV inside RotorS. The left side describes the simulation parts by RotorS and Gazebo and the right part describes the components found on a real MAV. Both parts should be as close as possible.

All components of real MAVs are simulated by Gazebo plugins and physics engine. MAVs are defined in RotorS in a modular way. They consist of a body frame, a fixed amount of rotors and some sensors attached to the body. All three type elements (body frame, rotors and sensors) are defined by its physical properties like size and weight. Some sensors like an IMU, a generic odometry sensor and a visual inertial sensor are already provided. To simulate realistic conditions, noise models for the provided sensors is available, in addition to the ground truth data. To facilitate the development of different control strategies, a simple interface is provided. Once, the MAV is flying, higher level tasks like obstacle avoidance and path planning can be tackled.

1) Packages structure: RotorS is split in packages that can be modified to add a new model and the main ones are: *rotors_description* that contains the URDF and other files defining the visual and physical properties of the MAV, *rotors_control* that contains the source code of the position and attitude controllers and *rotors_gazebo* that contains the launch and configurations files to start the simulation. Fig. 2 contains a diagram with all the packages available in RotorS.

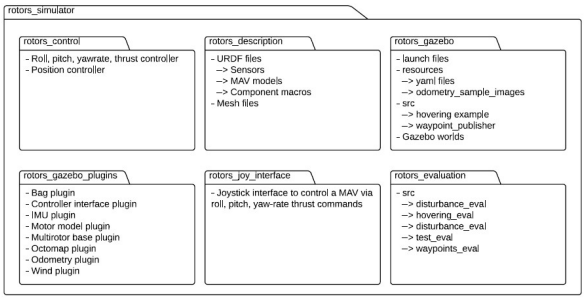


Fig. 2: Structure of the packages contained in the RotorS simulator.

III. RotorS: integrating Tyrion

In this section, we present the process of developing a realistic simulation model of Tyrion using RotorS framework and Gazebo.

A. Building a model

In order to have a realistic simulation, the first step is to have an exact replica of the body frame of Tyrion, with the real physical properties like mass, size or inertia.

We started modeling an exact replica of the 3D mesh of Tyrion (Fig. 3) using Blender [9].

(a) Real Tyrion (b) Virtual Tyrion

Fig. 3: Side to side comparison between an image of Tyrion from reality and an image from the 3D model of Tyrion

Next, we have created a model for Tyrion following the convention of the RotorS framework, which defines MAVs at three layers using Universal Robots Description Framework (URDF), adding components on top of each layer [1]:

1) Body frame and rotors: [2] this is the most simple layer. It instantiates the hexarotor body, defining the *base_link* frame and attaching the six rotors to the body frame using URDF joints. It defines the mesh to use for visualization and to define the dimensions of the MAV, as well as other physical properties like mass, inertia, arm length and rotors

[1]Files defininning Tyrion model are inside *rotors_simulator/rotors_description/urdf*

[2]Files at this layer are named *mav_name.xacro* (tyrion.xacro)

constants. All these properties have been defined according to real measurements except for the inertia matrix which has required a more extensive work. For calculating the inertia matrix a FreeCAD model [10] (see Appendix for more information), similar to [11], has been built but further work would be needed to finetune the values. There exists other experimental ways of calculating the inertia matrix like using a rotational pendulum [12] or an Unscented Kalman Filter processing flying data [13].

2) Adding an IMU: [3] Starting on top of the previous layer, it adds a controller interface, a mavlink telemetry interface and an IMU attached to the *base_link* frame. At this point, the MAV is ready to flight thanks to the attitude estimation given by the IMU, but it would not be able to estimate its position and, therefore it would not be able to perform any high level task autonomously. For this, as explained in Section II-A.4 we need some kind of sensors to estimate the position of the MAV.

3) Mounting sensors: [4] In the real world, Tyrion mounts a Pixhawk 4 autopilot [14] that contains an IMU, which we have added in the previous layer. Regarding sensors, Tyrion has a Realsense Tracking Camera T265 [15] for SLAM and a Realsense Depth Camera D435i [16] for understanding the surroundings and doing obstacle avoidance. Both types of cameras have been integrated into the simulation model using public libraries that simulate the sensors in Gazebo [17]. Tyrion has an additional GPS sensor that could be easily added to the simulation model thanks to the modular architecture of RotorS, where adding new sensors can be done by just modifying the XML file where the sensors are mounted. For our specific project of indoor navigation the GPS sensor is not a very important one since GPS data might not be available in indoor scenarios.

Our simulation model of Tyrion allows us to replicate reality quite precisely. Fig. 4 shows the comparison between real data being captured by Tyrion (Fig. 4a) and the simulation data captured by Tyrion inside Gazebo (Fig. 4b) using the sensors described in this section.

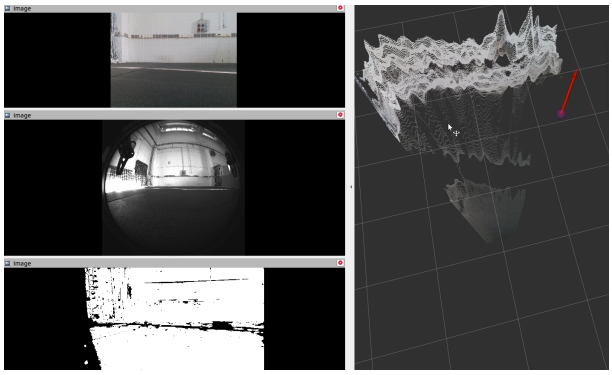
IV. Analysis and Validation

In order to evaluate the accuracy and correct operation of the sensors of our simulation model of Tyrion in Gazebo we have recreated an indoor house scene (Fig. 5) and used the position controller included in RotorS to follow a predefined trajectory inside the house. The trajectory is composed of four points and the objective is to reach the points as quickly as possible without overshooting when the points are published (Fig. 4b shows the sensor data at the final point). The same predefined trajectory (Fig. 6) has been performed by three different versions of Tyrion with different physical and control parameters for a more complete analysis.

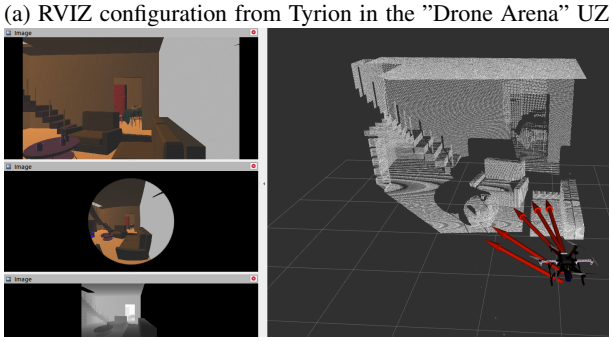
The first version is the original Tyrion simulation model with the original position gain value of the controller (Fig. 6a). We can see how the navigation to the points is done in a reasonable time and without much overshooting. We can also see how there is a little deviation in the movement of the drone in the y coordinate. *Note: The dimension of the y coordinate plot is very small compared to the scale of the x and z coordinate plots.*

The second version is the original Tyrion simulation model with a lower position gain value of the controller (Fig. 6b). We can see how the navigation to the points is done slower than in the first version due to the lower position gain value which makes the MAV move at lower velocities. There are no signs of overshooting and, similar to the first version, there is a little deviation in the y coordinate plot.

The third version is the original Tyrion simulation model but increasing the mass while maintaining all the original



(a) RVIZ configuration from Tyrion in the "Drone Arena" UZ



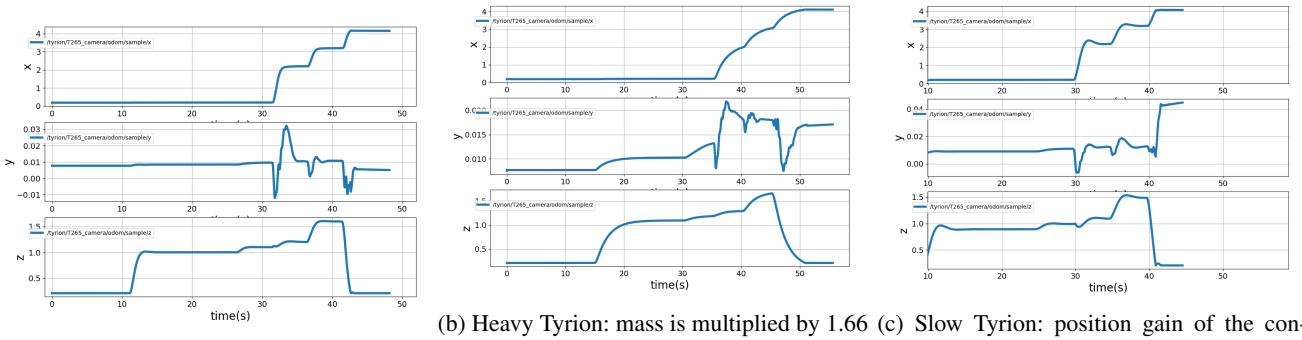
(b) RVIZ configuration from Tyrion in Gazebo

Fig. 4: RVIZ comparison between sensor data from reality and simulation. In both figures, the left part from top to bottom show an RGB camera, a fish eye camera for visual tracking and a dpeth camera. The right part shows a cloud point.

Fig. 5: 3D Scene "Tyrion house" used for testing

[3] Files at this layer are named *mav_name_base.xacro* (tyrion_base.xacro)

[4] Files at this layer are named *mav_name_with_sensorname.gazebo* (tyrion_with_t265_d435i.gazebo)



(a) Original Tyrion configuration

(b) Heavy Tyrion: mass is multiplied by 1.66

(c) Slow Tyrion: position gain of the controller is divided by 3

Fig. 6: Results for X Y Z coordinates for 3 tests performed with Tyrion. For more information about hte 3 tests see Appendix. *Note: scale of Y coordinates is different from X and Z*

position gain value of the controller (Fig. 6c). We can see how the navigation of the points is done in a similar time than in the first version but the overshooting is greater, which shows that if a physical parameter of a MAV is changed, like the mass of the hexarotor, all the other control parameter values must be changed accordingly. The deviation in the y coordinate plot is the biggest among all the experiments, especially at the last point (landing point), because it has been an abrupt landing.

V. Future Work

After integrating Tyrion inside RotorS, we have identified several interesting lines of work that could be followed to increase the functionality of the simulation model:

- For a more accurate outdoors positioning system, Tyrion could benefit from adding a GPS sensor. This sensor combined with the tracking system should provide a great estimation of the position and odometry. Also adding a more powerful and accurate LIDAR should be beneficial for navigating indoors in low light conditions where the tracking camera cannot provide good odometry.
- Once Tyrion has all the sensors mounted, we could do an integration with Octomaps for a DWA (global and local planners) to perform motion planning and collision avoidance. This would give the drone great capabilities for navigation in dynamic environments.
- Further and more extensive parameter fine-tuning is needed to check if the virtual model behaves correctly and in the same manner as the real drone. All control values (PID), masses and moments of inertia are susceptible for further fine-tune to match the simulated system with the real one.
- We think that a possibility for validating the real model with the simulated could be using a VICON system for motion capture. This could provide the real ground-truth of the real drone and this could be compared to the simulated.
- In this work we used the position controllers from Lee, but it should be checked if this suits the real controller on the drone developed by PX4. For future works it would be interesting to match the controllers from the real and simulated drones, so that the final migration to the reality is done as simple as possible.

VI. Conclusions

In this project, we have developed a simulation model of a real hexarotor available at RoPeRT - Universidad de Zaragoza for RotorS.

We introduced the problem of testing new algorithm on MAVs in the real world and the wide variety of disadvantages it has. We presented an overview of the theoretical background on the forces and dynamics used to model a MAV and an overview on RotorS, a framework for simulation models of Micro Aerial Vehicles (MAVs) in Gazebo.

We detailed the steps followed to create a simulation model of a MAV inside RotorS, from the 3D model to the definition of sensors mounted on the hexarotor.

We validated our simulation model with a trajectory navigation task in a indoor house scene and different variations of physical and control parameters, analysing the different behaviors. Special attention was given to the accuracy of the sensors mounted on the simulation model compared with reality.

Throughout this project, we have deepened our knowledge of MAVs, both from a physical point of view identifying different parts and sensors of the hexarotor as well as from a dynamic and control point of view. We also extended our knowledge on a wide variety of ROS related tools like rosbag, rqtree, RViz and Gazebo that have been seen in the laboratories during the Autonomous Robots course. Finally, we demonstrated to be able to use an open source large project by migrating a real hexarotor into into RotorS, a research oriented framework for modeling MAV. We would like to emphasize other transversal abilities gained during the project like reading scientific papers on robotics or the generation of a FreeCAD model to calculate the inertia matrix of the hexarotor.

References

[1] “Gazebo,” https://gazebosim.org/home, accessed: 2023-01-17.

[2] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26054-9_23

[3] K. V. Rao and A. T. Mathew, "Dynamic modeling and control of a hexacopter using pid and back stepping controllers," in *2018 International Conference on Power, Signals, Control and Computation (EPSCICON)*, 2018, pp. 1–7.

[4] H. Mazeh, M. Saied, H. Shraim, and F. Clovis, "Fault-tolerant control of an hexarotor unmanned aerial vehicle applying outdoor tests and experiments," *IFAC-PapersOnLine*, vol. 51, pp. 312–317, 01 2018.

[5] S. Rajappa, M. Ryll, H. H. Bülthoff, and A. Franchi, "Modeling, control and design optimization for a fully-actuated hexarotor aerial vehicle with tilted propellers," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 4006–4013.

[6] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[7] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5420–5425.

[8] 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," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

[9] "Blender," https://blender.org/, accessed: 2023-01-17.

[10] "Freecad," https://www.freecadweb.org//, accessed: 2023-01-17.

[11] A. Chovancová, T. Fico, Ľuboš Chovanec, and P. Hubinsk, "Mathematical modelling and parameter identification of quadrotor (a survey)," *Procedia Engineering*, vol. 96, pp. 172–181, 2014, modelling of Mechanical and Mechatronic Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877705814031981

[12] L. Derafa, T. Madani, and A. Benallegue, "Dynamic modelling and experimental identification of four rotors helicopter parameters," in *2006 IEEE International Conference on Industrial Technology*, 2006, pp. 1834–1839.

[13] A. Chovancová and T. Fico, "Design of a test bed for parameter identifation of actuator used in quadrotors," *ELITECH'14: 16th Conference of Doctoral Students*, pp. 1–6, 01 2014.

[14] "Pixhawk 4," https://docs.px4.io/main/en/flight_controller/pixhawk4.html, accessed: 2023-01-17.

[15] "Realsense tracking camera t265," https://www.intelrealsense.com/depth-camera-d435/, accessed: 2023-01-17.

[16] "Realsense depth camera d435," https://www.intelrealsense.com/tracking-camera-t265/, accessed: 2023-01-17.

[17] "Realsense gazebo description," https://github.com/m-tartari/realsense_gazebo_description, accessed: 2023-01-17.