A High Fidelity Simulator for a Quadrotor UAV using ROS and Gazebo

Mengmi Zhang*, Hailong Qin†, Menglu Lan*, Jiaxin Lin*, Shuai Wang*, Kaijun Liu*, Feng Lin†, Ben M. Chen†
*Department of Electrical and Computer Engineering
National University of Singapore
117576 Singapore
Email:{a0091624, a0091733, a0083022, a0092174, a0091844}@u.nus.edu
†Temasek Laboratories
National University of Singapore
117411 Singapore
Email:{mpeqh, linfeng, bmchen}@nus.edu.sg

Abstract—Flight tests of prototype UAV systems can be restricted by spatial constraints and they may bring risks of damage due to failures. Motivated by these, we presented a simulation approach based on Robot Operating System (ROS) and Gazebo. Unlike other state-of-the-art quadrotor simulators, we implemented the dynamics model of the UAV in ROS to achieve high fidelity behavior of the UAV. A hierarchical navigation system is also presented in our paper. The system layers include simultaneous localization and mapping (SLAM), mapping framework in Cartesian and polar coordinates, A* global path planner, revised vector field histogram plus (VFH+) for optimal local path selection and online trajectory algorithm (OTA) with collision checking for obstacle avoidance. In order to cater for vision-based applications, quadrotor is equipped with a monocular camera in the simulation model. The implementation of circle and landing pad detection and tracking algorithm demonstrates the functionality of vision guidance. In our simulation, various aspects including complex indoor and outdoor environments and on-board sensors are capable of simultaneously interacting with our navigation system to achieve certain surveillance missions. In the end, we demonstrated the applicability of our complex quadrotor systems by performing an autonomous navigation task in simulated complex environments. In comparison with the experimental data, simulation results align with the ones in flight tests in terms of real flight behaviors during navigation tasks in general.

I. INTRODUCTION

Aerial robotics have been widely studied in potential applications, such as surveillance [15], navigation [17] and transportation [12]. Despite conventional helicopters, a multicopter is a newly emerging form of aerial vehicles. Due to their simplicity in structure and maintenance, they have been widely employed by researchers to study autonomous capabilities of unmanned systems in challenging environments. However, testing prototype quadrotors under realistic scenarios can be effort-intensive and failure of the testing algorithms may lead to damage. Simulation environment provides an effective tool to help researchers with the ease of reconfiguration, the availability of simulated hardware components such as onboard sensors and the flexibility of environmental setup.

Researchers used to simulate quadrotor UAV systems using commercial tools like Matlab/Simulink [11] or some flight simulators such as FlightGear [6], X-plane10 [8] and GEFS [7]. Although these tools provide simulation frameworks, they still require significant amount of efforts in testing sensor-based high level control [13]. Besides, some commercial simulators only lend restrictive modifications and enhancements to users. Due to various limitations, ROS-Gazebo provides a standard simulation framework based on its modular architecture in robotics research and it facilitates integration of contributions by other researchers. One of the integrated flight models in ROS Gazebo is contributed by the research group from TU Darmstadt [13] with the focus on incorporating external disturbances such as wind into quadrotor dynamics. There are also subsequent works [14] [2] on simulating quadrotor landing on dynamic platforms based on Meyer’s quadrotor model [13]. In navigation system simulation, Miguel proposed an optimized fuzzy visual servoing system for obstacle avoidance for quadrotors in ROS Gazebo. To our best knowledge, none have simulated a dynamics model for quadrotor UAVs in ROS and integrated with a hierarchical navigation system to complete a complex testing course autonomously with high fidelity of UAV behavior, sensor model and environment so far.

As a result, faced with those challenges, our group develops research interests in developing a simulator in ROS Gazebo to test the quadrotor behavior and its navigation capabilities. In our paper, a complex indoor and outdoor course is designed and the environmental model is imported into Gazebo. To cater for all these various navigation tasks, a quadrotor model with an integrated sensor set is proposed. Universal robotic description format (URDF) model of our quadrotor is also configured in ROS in Section IV-A. As one of the unique features of our simulator, the robust perfect tracking (RPT) based control dynamics of our quadrotor model is introduced in Section III. The control model has been developed by our research group for the past few years. As a separate
control model built in as a ROS node with quadrotor position, velocity and acceleration as control inputs, all these states’ control outputs can be generated after heuristic outer and inner loop controls. This encapsulated ROS node enables us to tune simulated control models with flexibility and handiness. The state of the simulated quadrotor model in Gazebo then follows the control output. Simultaneous interaction of sensing and navigation functionalities including SLAM, path planning and vision guidance can be carried out. New references to the controller are generated. The overall navigation system and the control model loops in the entire process from the quadrotor take-off to the safe landing. Both comprehensive navigation system of the quadrotor and the implementation of our simulation system are presented in Section IV-B and Section IV-C respectively. In the end, a real flight test was conducted. By comparing with the simulation results and real flight test results, we evaluated the performances in Section V with conclusion and future work in Section VI.

II. Background Information

ROS was originally developed in 2007 at the Stanford Artificial Intelligence Laboratory [11]. With its open source contributed from diverse research communities all over the world, it is embedded with many handy libraries and visualizers designed specifically for robot applications. ROS is flexible due to its modular architecture. Same nodes can be repetitively used in various simulations. Gazebo is equipped with high-performance physics engines, realistic rendering environments, sensor plugins with the option of contaminating measurement data by noise and integration of customized plugins for robots, sensor and environment. Our simulation system is built based on open dynamics engine (ODE) in Gazebo which specifies timing, gravity, friction, collision and other rigid body dynamics. Gazebo can interact with ROS via node communications. Each sub navigation module is encapsulated as individual node subscribing necessary inputs and publishing outputs as topics.

URDF is a standardized XML format file in ROS used to describe all robot elements. In URDF, users can specify the kinematic and dynamic properties of a single robot. It heavily relies on XML descriptive languages. In order to prevent humber drum manual coding process, we encapsulate variables sharing the same properties in Xacro package such that they can be treated as constants, arithmetic computations or commonly used packages. collision and visual elements are different in Gazebo. Gazebo will only treat the elements in collision as “visible” to laser scanners and collision checking. Sometimes the robot model can be complex and its collision checking can be computationally expensive. Hence, we strongly recommend to use simplified collision model. After setting up each element correctly, a URDF parser can be used for checking syntax errors before the URDF model is loaded in Gazebo.

The proposed simulator follows standard signal and logic flow shown in Figure 1. 3D testing environment model can be imported from 3rd party softwares. Sensor set configured in robot URDF file interacts with the environment simultaneously publishing raw data. After a systematic way of information processing in navigation system, new control reference will get published to high fidelity control dynamics simulation node. The dynamic behavior is simulated and reflected in Gazebo environment based on ODE. This loop continues until user-defined termination commands are issued.

III. Modelling and Control of the Vehicle

Before we introduce more on how to model and control the vehicle, we would like to define the coordinate frames used in this paper. Different from the world frame in Gazebo, the global frame in our navigation system is based on North-East and Down (NED) frame. The map updating and path planning modules are according to NED frame. The origin of the body frame is the center of gravity of the vehicle.

The overview of the system structure is shown in Figure 2. The reference commands \( (\delta_{ail}, \delta_{ail}, \delta_{ail}, \delta_{ail}) \) are fed into inner-loop control of the quadrotor. In the outer-layer, the quadrotor heading \( \psi \) is the integration of \( r \), and its vertical axis position is the integration of \( w \). As \( (\phi, \theta) \) angles will induce acceleration along x and y axis in the body frame, they can integrate and obtain the velocity vector \( (v_x, v_y) \) in x and y axis body frame and transfer to NED frame. Velocity can further integrate to obtain the position \( (x, y) \).

As a quadrotor model with an inner-loop controller NAZA is highly symmetrical (Naza is an off-the-shelf all-in-one attitude controller designed for multi-rotor flying platforms), it is reasonable to assume the roll and pitch motions are the same. The model can be decoupled among all four channels independently. The model identification process is done by analyzing the frequency response from input and output data pairs and hence fitting its transfer function, which has been detailed in [18]. Roll and pitch channels have been described.
by the following 4th-order transfer function:

\[ \frac{\phi(s)}{\delta_{ail}(s)} = \frac{\theta(s)}{\delta_{ele}(s)} = \frac{9688}{s^4 + 27.68s^3 + 485.9s^2 + 5691s + 15750} \]  

Roll pitch angle to longitudinal velocity can be described by the following first-order transfer function:

\[ \frac{v_x(s)}{\phi(s)} = \frac{8.661}{s + 0.09508} \]  

Yaw channel can be defined as follows:

\[ \frac{\psi(s)}{\delta_{rud}(s)} = \frac{3.372}{s} \]  

Heave channel is given below:

\[ \frac{w(s)}{\delta_{thr}(s)} = \frac{-13.35}{s + 2.32} \]  

In order to numerically solving and discretizing approximation of solutions of the above ordinary differential equations, we adopted Runge-Kutta Method [16].

After identifying the dynamics model of the vehicle with the NAZA attitude controller, we will design the outer-loop control law to track a 3D position and heading reference. The outer-loop control signals are all defined in NED frame and for all three directions, the dynamics are approximately formulated as double integrators. Since the translational motion in these three directions are almost decoupled (inner-loop should have decoupled them if designed correctly), the control laws for these three channels can be designed separately.

We propose in this paper the design of the outer-loop controllers using the so-called robust and perfect tracking (RPT) control technique that resulting closed-loop system is asymptotically stable and the controlled output almost perfectly tracks a given reference signal in the presence of any initial conditions and external disturbances given a system that satisfies certain conditions [4]. Such capability is extremely desirable for flying vehicles involving complicated maneuvers. Since the outer-loop dynamics are all standard second order systems, by choosing an appropriate natural frequency and damping ratio, the vehicle should be controlled without any problems. Of course, minor tuning is needed after several flight trials have been conducted. RPT control for quadrotor has been developed pretty mature by our research group for the past years, and interested readers please refer to [18] [3] [9] for detailed design procedure.

IV. IMPLEMENTATION OF SIMULATION PLATFORM

A. Physical Models in Simulation

In order to test the full capabilities of the quadrotor prototype, a specific simulation environment is setup shown in Figure 3 compared with the testing field in reality. The model was designed in 3DS MAX and imported into Gazebo using URDF language where its base link is now attached to the fixed world frame. The mission includes autonomous take off, fly through a circular window with the width of 1.5m, avoid objects such as a fan, fly through a corridor, release payload at an unknown target location which is indicated by concentric circles, fly through a door, capture information on the LED screen, clear the pole arrays which simulates the forest environment, land on the landing pat at an unknown location.

For simplicity, the entire quadrotor is modeled as a box block for collision element. In order to have better visualizations, we imported the mesh file of our quadrotor model in STL format and attached all the sensors to the body as additional links. Currently two laser scanners were used with the one below fixed for altitude control and 3D map construction while the other is attached on the top for navigation purpose. One monocular camera is attached to a tilting mechanism for circle detection and vision guidance. The tilting angle varies from \([0^\circ, 180^\circ]\) with its resolution 1^\circ.

Measurement data can be contaminated with noise. Here we assume the noise follows Gaussian model in general. In Section III, measurement noise from the accelerometer and the gyroscope can also be taken into account. These noise is directly introduced into the measurement as control inputs.

B. Navigation System Overview

As we aim at a comprehensive approach for simulating our quadrotor, the navigation system in particular, an overview of
our navigation system is presented without going through all the rigorous algorithm analysis and design reasons behind. Based on our designed mission in Section IV-A and payload constraints, we attached a quadrotor structure with two laser scanners and one camera. We assume all our trajectory generation algorithms are based on 2D maps and hence a constant altitude maintenance is desired. One 2D LIDAR is positioned horizontally and its measurement data will be used for SLAM and path planning. In order to obtain a 3D map of the environment, another LIDAR is oriented toward the ground. As the quadrotor moves around, its 2D scanning region contributes to individual layer and hence forms a 3D octomap. Besides, the scanning data can be processed further for altitude maintenance. The measurement data has to be synchronized with the current pose of quadrotors. All the accumulated LIDAR data can then be used for updating 3D octomap which is to be transmitted to ground station for real time visualization. Due to weight limitation of the quadrotor, one monocular camera is employed on a position-based servo motor for both vision guidance and surveillance mission. It is initially oriented downwards for ground target detection. As the quadrotor is guided by the ground target location computed from the image, reasonable control command is issued to the tilting mechanism such that it ensures the ground target is always within visible range. In other words, the target of interest is always on the image. Since the monocular camera can only provide 2D information, the coordinate of the ground target in the global frame can be obtained with additional information, i.e. the flying height above ground of the quadrotor itself. After payload dropping mission is accomplished, the camera is facing backward for information searching on the LED screen. Altitude heading reference system (AHRS) is good within short time period and provides fast response. However, it is not suitable for long-term reference due to the drift from the accumulated errors during integration. Its measurement data will be fused with the updated pose from SLAM in order to achieve better control stabilization.

Specifically, the diagram of path planning algorithms is shown in Figure 4. In the unknown environment, quadrotor is programmed to simultaneously estimate its pose and update 2D maps. We assume quadrotor is moving in a horizontal plane with a constant altitude. This is due to the constraint of 2D SLAM[10]. Though only 2D scanning data points are used for pose estimation, 3D laser data cloud can still contribute to updating the octomap only for surveillance purposes. Within the partially discovered area, a global path planner using A* can return a global path from the current position to the target while treating all unknown regions to be free temporarily. Only the line-on-sight segment of the global path will be the active region for trajectory generation. The newly computed obstacle free smooth trajectory can then pass to Reflexxes Motion Library. Via an online trajectory algorithm and current motion state of quadrotors as the function input, it will be able to return an appropriate trajectory applicable for lower level controllers. By linearizing the estimated trajectory, each line segment have to pass collision checking tests in order to make this trajectory valid. If this optimal path has failed, the revised VFH+ system will be reactivated and find the second optimal path which will do collision checking again. This algorithm will keep searching for the optimal paths until one is confirmed to be obstacle free. This whole process continues until quadrotors arrive at the destination. This systematic way of planning path was developed and tested in our research group[19].

C. Simulation Structure Overview

In ROS and Gazebo, we implemented our individual navigation modules as nodes and interacted with one another as shown in Figure 5. From the system overview, we have one monocular camera and two laser scanners. We construct quadrotor URDF model with all sensor plugins. Detailed explanation on its physical model is explained in Section IV-A.

In order to control the tilting vision unit, we attached an effort joint servo motor and it is based on position control. Here we adopted a simple PID controller with triangular function as reference input. Node MoveCamera sends position control commands at a rate of 50 Hz. The position command will publish corresponding joint information as jointstates. The node RobotStatePublisher manages the update of all the relevant robot link transformations. It will publish the updated link information into tf which is a coordinate system management center in ROS. All the sensors interact with simulated environment in Gazebo and obtain the measurement data by publishing them to different scanning topics. TaskManager is the headquarter node for sending trajectory commands and organizing interactions among different modules. Controller performs quadrotor dynamics simulations. In the end, it will publish its desired pose information to the global frame via calling service SetModelState in Gazebo and updating this model state to tf. PathPlan subscribes the scanning data topics, updates map and finds optimal trajectory according to the waypoint commands TaskManager issues. CircleDetection is in charge of circle detection and tracking. Together with AltitudeControl which is used for altitude information ex-
traction, CircleDetection returns TaskManager with the location of the ground target in the global frame. HDetection is another node similar to CircleDetection. HDetection does not only require the accurate detection of circles but also detection of the capital letter H on the landing pad. For better visualization and real-time map construction, we use the very convenient tool RVIZ from ROS. The recorded trajectory coordinates get published to topic VisualizationMarker and can be used for displaying the quadrotor path as a thread of point markers in RVIZ. The scanning data from the LIDAR unit pointing downwards can be fed into the node Scan2PointCloud which converts scanning data into type PointCloud in ROS based on the global frame and publishes these data as the topic MyCloud. Node Octomap can then construct the octomap and RVIZ use the topic OccupiedCellsVisArray for displaying the occupied nodes in the octomap which is a structure explained in [5].

Due to the hardware limitations, threading is a serious problem where synchronization of data is compulsory. Fortunately, ROS provides us a very convenient platform for tuning threading parameters such as buffer size for data storage and the running rates for various nodes and sensor drivers. To match with the hardware specifications, buffer sizes and running rates for individual nodes are the same for both simulation and flight tests. In our paper, we used the on-board microcomputer Asctec Mastermind 3rd generation from Ascending Technology for high level computations such as computer vision and navigation stacks. It is equipped with high performance Intel Dual Core i7 processor with 4GB RAM. Another on-board microprocessor is Overo Fire COM from Gumstix Inc and it is mainly used for the flight control. The communication between these two microprocessors has to be fast and synchronized. ROS programs are run on Groovy version. All the buffer sizes and node update rates are listed below in Table I based on the platform we just introduced.

### Table I

<table>
<thead>
<tr>
<th>Nodes or Drivers</th>
<th>Buffer size</th>
<th>Update Rate (HZ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToGumstix</td>
<td>1000</td>
<td>20</td>
</tr>
<tr>
<td>SLAM</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Camera Driver</td>
<td>1</td>
<td>1.875</td>
</tr>
<tr>
<td>Hokuyo Laser Scanner 30m</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>Hokuyo Laser Scanner 4m</td>
<td>100</td>
<td>5</td>
</tr>
<tr>
<td>Circle Detection</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Landing Pat Detection</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Path Planning</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Gumstix</td>
<td>1000</td>
<td>50</td>
</tr>
<tr>
<td>Computer Vision Flag</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Servo Control</td>
<td>100</td>
<td>20</td>
</tr>
<tr>
<td>Height Measurement</td>
<td>100</td>
<td>5</td>
</tr>
</tbody>
</table>

V. RESULTS AND DISCUSSION

A flight test in the simulation environment was conducted. Figure 6 shows the constructed octomap after quadrotor successfully navigates through the testing course and arrives at the destination. As the update rate of the laser scanner is low, the octomap cannot fully cover all the places. However, it matches with the simulation environment we set up before.

Figure 7 below shows the final result of the SLAM algorithm. We employed the FastSLAM algorithm in indoor navigation tasks and GraphSLAM in outdoor ones. As our quadrotor is assumed to maintain at a constant altitude, only the positions in x and y direction in the global frame are plotted with its orientation θ. The trajectory of the quadrotor is shown in the first sub figure. Though there are some discrepancies in poses between the ground truth and the estimated, the error lies within acceptable range, less than 1m. The possible explanation for relatively large drifting errors in the x direction are due to the lack of wall or tree trunk features in the end. The quadrotor is not able to locate itself without enough features as it flies out of the outdoor pole arrays. From the graph, we can also see that the estimated orientation matches perfectly with the ground truth.

As the 2D costmap is crucial to the path planning algorithms, a global costmap is plotted on the fly. It is verified that those regions which are nearer to the obstacles will have
higher costs which push the quadrotor away during trajectory generation. The brighter colors indicate higher values of cost. Different from the simulation environment, in order to ensure the quadrotor fly within the confined region without surrounding walls, dummy square walls on four sides are added in the costmap.

From the overall position plots of both simulation results as well as flight test results in Figure 8, the measurement generally follows the reference though discrepancies do exist. There are various reasons, e.g. the simulated control system is assumed to be ideal whereas this is not the case in reality as more noises including measurement noise, external current disturbances, and control noises are introduced into the system. As it is a dynamic process, our developed control models will not be able to handle perfectly well with places such as narrow corridors and dense pole arrays. Besides, the localization information obtained from SLAM also introduced errors to the system. Nevertheless, the deviations from the measurement to the reference lie within the tolerable range roughly to be ±0.5m. Path planning algorithms will ensure the quadrotor flies through an obstacle-free path by various collision checking algorithms and optimal trajectory generation approaches.

VI. CONCLUSION AND FUTURE WORK

The research work contributes toward the simulator for a quadrotor UAV using ROS and Gazebo. In our paper, we present a comprehensive simulation platform by introducing high fidelity flight dynamics simulation. Currently, we manage to set up testing platforms in ROS and Gazebo including basic UAV physical and dynamic models and simulation environments. We also carry out a complex simulation test where UAV can autonomously complete both indoor and outdoor courses and arrive at the destination by integrating our developed navigation system into the simulation environment. The simulation results are reasonable based on theoretical analysis. They have also been validated with greater details in terms of signal responses, behaviors and control states of our quadrotor after comparing with the real measurement data during flight tests. As simulation is a very useful tool for prototype testing, next we are going to integrate ROS with Unity 3D platform for more realistic simulation needs.

REFERENCES