Integrated Software Development Methodology for Real-time Precise Control of UAV

Jung Kyu Park¹, Eun Young Park² and Heung Seok Jeon³*

¹Dept. of Computer Software Engineering, Changshin Univ., Changwon-si, Korea
²Dept. Biomedical Laboratory science, Shinhan University, Uijeongbu-si, South Korea
³Dept. of Computer Engineering, Konkuk University, Chungju-si, Korea

*Corresponding author

Abstract. Recently, drones have been widely used in education and industry. In this paper, we propose a navigation and control algorithm to develop unmanned drones quickly. To limit the proposed framework, we analysed the existing drone solution and simulation framework. We also used Hardware-in-the-Loop (HIL) simulation for rapid development. HIL will simulate physical systems that interface with embedded control equipment on real-time hardware. The code generated in this process can be experimented on a hardware device so that the control performance of the drones can be verified without draining the drones in the outdoor environment. Using the proposed method, you save time and money on drone development while providing stability.

1. Introduction

A major issue in the automotive market in recent years was the introduction of autonomous driving technology. From three years ago, self-driving cars have been steadily increasing. Many research institutes are predicting that unmanned vehicles will replace people's vehicles. This technical success can provide many benefits for our lives. The use of this technology also provides safety and efficiency in operations [1].

As mentioned above, the autonomous system can be used in situations where it is difficult for humans to work or in extreme environments where it is impossible to work. Autonomous systems use the latest sensors to process events using collected information, so they can handle more events than people in the same place. For example, in a surveillance system, autonomous systems are more efficient because they can be processed faster by using more information than humans. In addition to the guard system, autonomous systems are being used in a variety of fields such as military operations, crop cultivation, and fire monitoring [2]. Nowadays, there are new problems to be solved as they are used in general fields besides special fields, and studies are being done simultaneously to solve problems.

It can be seen that the representative automobile of autonomous driving is made by the development of new sensor due to the development of IT technology. For example, in the field of animation, you need to be able to bring new advances in the computer and graphics technologies to the market. For this reason, it is recommended that new frameworks be used to exclude existing frameworks in order to develop and validate new research systems. The most useful tool for system development and performance evaluation is a simulation environment for quick feedback [3].
An autonomous vehicle equipped with a computer for automatic driving can be said to be an embedded system [4, 5]. Unlike general-purpose computer systems, embedded systems are equipped with CPUs and sensors that are specially designed hardware and software that performs control functions and is used for control functions (Figure 1). These embedded systems require special requirements to obtain simulation experiments and verification results. In the general case, when you try to develop a specific algorithm, it is a real requirement to implement a real simulation environment. However, in the embedded field, there are many things that are impossible with the previous implementation. It is important to keep in mind the computer hardware and software capabilities that perform the operation and control of the physical part of the actual system, which may have produced accurate results.

In the simulation environment, the embedded system in Figure 1 can be divided into two areas. At the top, the control system is responsible for control, while the bottom sensor, the actuator, and the environment are responsible for the infrastructure. Simulations can be viewed as data flows in these two areas. In the lower area, the sensor data input from the outside is input and output. In the upper area, the calculation for the control is performed using the given sensor data.

Two simulation methods can be used when the theoretical content is applied to the actual implementation in the control system. The first is the Software-in-the-Loop (SIL) and the next is the Hardware-in-the-Loop [5, 6, 7].

The SIL runs within the computer and consists of programming code. In other words, all simulation elements (input data and output, control) are all simulated within the computer system. To perform this simulation correctly, you need to take the code that runs on the embedded system and run the experiment on a system that can handle this code. This is called SIL because the code responsible for control is part of the software. HIL, on the other hand, uses both the computer that performs the simulation and the hardware device on which the algorithm code is run. The hardware device responsible for the control is called the HIL [8].

The advantage of SIL is that it does not require the expensive embedded board to be tested, and there is no board damage. It also has the advantage of being able to debug easily since it is experimented in the computer. However, HIL has difficulty debugging because it has to conduct a real hardware experiment. The disadvantage is that it is not an experiment on a computer such as a board performance, so it is difficult to predict the performance on an actual board. For this reason, you should know exactly what your application needs when setting up your experiment. SIL can also be useful because of its ease of implementation. HIL can be more difficult than SIL and requires accurate understanding of the hardware, but can be used for accurate performance measurements.

In this paper, HIL simulation was used [9]. This is because the HIL provides an environment for developing new algorithms targeting real hardware and for evaluating performance. While SIL simulation can be performed more easily than HIL, it can be difficult to conduct experiments.
considering the actual environment. For example, SIL cannot catch various bugs, such as jitter bugs or bugs that only appear in certain situations, as they do on hardware.

2. Software System Architecture for UAV

As mentioned earlier, when designing hardware using HIL simulation, various components should be considered. In this environment, the application must be installed on the simulator computer, and the user interface application must be connected to the embedded board. The paper consists of a workstation and a drone board that can perform various experiments. Figure 2 shows the relationship between the internal elements in the proposed system.

![Figure 2. System architecture](image)

The new idea in the proposed method is to split the data flow into two channels using the autopilot board. The data flow associated with the SIL must be of high priority because it simulates "real world" information representing the real-time flow. However, there is little real-time constraint on the information the user communicates.

The two data flows are processed by specialized data flux program. This program is written in C++, uses ptask to perform timing analysis and uses the results to evaluate system performance. We also used the FlighthGear Flight simulation program to simulate the virtual environment throughout the system. In the internal design, UDP socket communication was implemented using Matlab / Simulink.

Through the implementation of the functions described above, it is possible to visually check the operation of the unmanned drones. The data transfer from the board to the simulation workstation uses the UART interface and the data majors between the programs running inside the workstation used UDP sockets. There are three reasons for using UDP without using TCP on your workstation.

1. No internal connection required
   Typically, TCP uses a three-stage handshake before sending data. This process has the problem of delaying data transmission. UDP is advantageous for internal data transmission because it does not use handshake method.

2. Data transfer overhead is small
   UDP is a transport layer connectionless protocol. Therefore, unlike TCP, there is no connection setup, and congestion control is not used, which is faster than TCP. Also, the segment header is smaller than TCP, which is 8 bytes and 20 bytes.

3. High transfer rate
   UDP has the advantage of faster congestion control than TCP. However, since there is no guarantee of data transmission, packet loss may occur. However, UDP detects a minimal error through the Checksum field in the header.
The basic UPD is a non-connection protocol, but it ensures the reliability of the connection through program implementation. In the following chapters a little more is explained [10].

2.1. Component of Simulation
Simulation components are a key part of the system that must reproduce the actual behavior. The components included in the system are shown in Figure 1. In experiments, system dynamics are represented by physical laws governing the motion of bodies in space. The sensor is mounted on the unmanned drones and the actuator is the four DC motors mounted on the drones.

We used Matlab / Simulink to implement the simulation components as described above. The reason for using Matlab / Simulink in the experiment is because of the problem with the original simulator. This program is used in many fields to model various systems. It also does not limit the time spent using the CPU to process data and other program elements. In the experiment, the FlightGear Flight simulation engine was integrated to meet the requirement to visualize the simulation state. Here are some reasons to use the FlightGear engine: First, this engine is open source and well documented. Second, a variety of API offerings enable you to quickly develop new systems. Third, it supports the integrated development environment, allowing easy debugging of programs. Fourth, it is easy to update APIs that users can use. Finally, the engine is updated periodically and can be reliably used.

3. Base Station
In this paper, QGroundControl program was used to implement Ground Station. QGroundControl is one of the solutions included in the Dronecode project under Linux Foundation [12].

The base station is a key component of the system because it communicates with the unmanned drones and the user interface is installed. This allows you to check whether the unmanned drones are operating or not. The base station allows you to filter the sensor values, determine the value of the Flight Control parameter, and pass specific commands to the unmanned drones. In QgroundContol, the user can specify the movement path of the unmanned drones as a drag on the map of the screen and load the contents into the autopilot. Telemetry data and sensor values that show the status of the drones can also be viewed directly on the screen and saved for replay.

Base stations using QGroundControl use the MAVLink protocol and support a variety of autosavigation boards such as PX4 Pro, ArduPilot, and pxIMU [11, 12]. It can also be run on a variety of operating systems, including Windows, Linux and MacOS.

QGroundControl is a software framework that helps you develop drone-related solutions as quickly as possible. This allows you to develop your program source in a modular fashion and make it easy to modify or add features later. Developers can easily incorporate new equipment into the system or easily modify the graphical interface using newly developed graphical tools. One of the greatest features of this base station is that it uses the MAVLink protocol to communicate with the unmanned drones. The MAVLink protocol is currently being used by many autonomous navigation boards and is open source, so it can be modified for development purposes [14].

4. Framework for Simulation

4.1. Simulation
The simulation component is responsible for two things. It carries out the actuator command created on the autopilot board and also provides the actual sensor data. The sensor data is input to the autopilot board and used for the simulation loop. Simulation components can be roughly divided into three parts.

1. Model of UAV Dynamics
   This part models the dynamics of an unmanned drones. And it calculates the orbit based on the actuator command input to the system.

2. Model of Sensor
   The sensors are mounted on an unmanned drone and provides the current status and external information of the drones. All sensor data is transmitted wirelessly to the base station.

3. Model of Environment
The environmental model describes the external environment in which the drones operate. The external environment includes current temperature, air density, and gravitational acceleration.

4.2. Model of UAV Dynamics

The UAV used in this study assumes a quad-copter with four rotors. In studies involving quadrotors, several methodologies have been proposed that relate to kinetic models for movement. Several studies have studied the model for detailed analysis and modeling [6-8]. As with these studies, advanced control algorithms are required to accurately describe the model. Especially if the UAV to be used is to be used without modification for a long time, a particularly accurate model is required. This study has no restrictions on specific UAVs with specific settings. We have chosen a very simple geometric model to develop dynamic equations for UAVs. This assumes that the proposed model can be applied to most UAVs. The main frame of the quadrotor can be built with a cross-shaped frame and can consist of four motors and a propeller of the same length. Due to the nature of the quadcopter, which uses four motors as propulsive forces, it is difficult to keep the entire level at all during initial operation and flight, since they affect each other as well as themselves. Therefore, the balance value can be implemented through PID control to achieve stable centered flight.

4.3. Model of Sensors

In order to fly the drone, information such as position, altitude, speed, direction, and obstacles are basically required. This information is essential for controlling unmanned aerial vehicles. It is also important that the drones fly safely and hover.

Most of our drones are equipped with 5 ~ 6 sensors basically. It is almost standard in consumer drones such as 3 axis accelerometer, 3-axisgyroscope, magnetometer, barometer, GPS sensor and distance meter.

The Matlab / Simulink programming environment provides many tools for developing various sensor models. Inside, there are various libraries available to use sensor blocks already developed and used. The sensor data used in the autopilot's algorithm to simulate the function simulator has been simulated.

1) Magnetometer

A magnetometer is a compass sensor. Measure the magnetic north and send the direction information of the drone to the drones’ CPU. This sensor is built in to drones with GPS capability. The movement of the drone can be grasped by combining the GPS position information, the magnetometer direction information, and the accelerometer movement information. The use of GPS drone is limited in this latitude above, since measurement of magnetic north is impossible at above 70 degrees north latitude. This sensor is basically a compass. In this reason, it is affected by magnetic objects in the vicinity. Electric power lines that generate electromagnetic waves, steel structures such as electronic devices and automobiles also have an effect. If a magnetometer error occurs, it can be resolved by moving slightly from the current position.

2) GPS Sensor

All aircraft use satellite GPS signals to measure the exact location and altitude of the aircraft. In recent years, as the GPS reception sensor has become cheaper, it has been installed in smart watches as well as hobby drones. The top GPS satellites were used for military purposes in the United States and Russia. Currently, however, it is open to commercial use and is used in various fields such as all aircraft, unmanned aerial vehicles, and automobiles. Of course, our GPS map and navigational map (NAVI) we use every day are also thanks to this GPS signal. In Matlab, the sensor model takes local coordinates and uses Simulink to convert latitude, longitude, and altitude.

5. Experiments

In this chapter, we measured the overall performance of the proposed system framework. In this experiment, we experimented with simulations at workstations, measured the time of simulation related processes, and analyzed the results according to time constraints. By analyzing the system time results, we confirmed that the system was operating correctly, and we identified the problem and identified areas that need to be corrected. Using the experimental data, we evaluated the meantime,
standard deviation time, best and worst time of a particular job. Through such mathematical analysis, we analyzed the performance of the proposed framework.

5.1. Experiment Setup
In the experiment to derive the results, the autopilot board was connected to the data management program and the workstation on which FlightGear was performed [13]. To ensure workstation performance, the Matlab / Simulink program was run on another computer connected to the local network. We also made additional connections with the board to monitor system status through the shell screen. We built the experimental environment and performed the simulation several times, logging the data for analysis. Statistical analysis was performed on the stored data using Matlab.

5.2. Experiment Results
Figures 3 and 4 show the execution time of each task. Analysis of the values reveals that the internal data flow is being delivered without significant delay during the operation. However, as can be seen from the graph, the standard deviation of about 3700us is large even if the average value is in the allowable range of the delay time. As shown in Figure 5, the internal-arrival time of the simulator's sensor data can well exceed the requested 4000us. Figure 6 shows the delay time over time during the simulation.

We analyzed the simulator response time data and found negative data. The time required to get a response from the simulation is greater than the 5ms sampling time on the autopilot board. The average response time is about 11ms and the standard deviation is 12ms. This problem can be expected to be a difference in the way Simulink implements UDP blocks.

**Figure 3.** Execution time of the inflow job

**Figure 4.** Execution time of the simulation job
6. Conclusions
The proposed UAV framework in the paper suggests a way to quickly develop navigation and control algorithms. Developers can run on the control board targeted by the algorithm they develop and see that it works correctly. This is because timing constraints can be met using timing precision. Considering execution time, the loop simulator can check the accuracy of the simulation flow itself to ensure the function of the hardware. Experiments have shown that the simulator generates a constant, non-negligible delay. This problem is related to the UDP implementation issue in Matlab / Simulink.

Acknowledgments
This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2018R1C1B5046282).
This work was supported by Changshin University Research Fund of 2019(Changshin-2019-32)

References
[1] Kyungbok S, Kyoungwook M and Jeongdan C. 2018 Proc. Int. Conf. on Advanced Communication Technology (ICACT) (Gangwon-do) (IEEE) pp. 300-302
[2] Kwangjae L, Heewook K, Tae Chul H and Jae Young A 2017 Proc. Int. Conf. on Information and Communication Technology Convergence (ICTC) (Jeju) (IEEE) pp. 1222-1224
[3] Prabhu Jyot S and Rohan de S 2018 Proc. Int. Conf. on Information and Communications Technology (ICOLACT) (Yogyakarta) (IEEE) pp. 168-173
[4] N. Nithya, P. Vinothkumar, Liza M K and S. Riyas Parveen 2017 Proc. Int. Conf. on Emerging Devices and Smart Systems (ICEDSS) (Tiruchengode) (IEEE) pp. 68-71
[5] Wang Ting T, Bai B, Yuan Yi F and Zhu Yong W 2018 Proc. Int. Conf. on Mechatronics and Automation (ICMA) (Changchun) (IEEE) pp. 1972-1977
[6] Rongxiao W et al. 2017 *Proc. Int. Symp. on Distributed Simulation and Real Time Applications (DS-RT) (Rome)* (IEEE) pp. 1-4

[7] Muhammad Faisal S, Ahmad B and Hasnain A 2018 *Proc. Int. Conf. on Applied Sciences and Technology (IBCAST) (Pakistan)* (IEEE) pp. 340-346

[8] Nkemdilim A O and Matthew C T 2016 *IEEE Transactions on Control Systems Technology (IEEE)* vol. 24, no. 6, pp 1980-1992

[9] ZeFang H and Long Z 2018 *Proc. Int. Conf. on Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) (Chengdu)* (IEEE) pp. 1254-1258

[10] Shuhang Z, Hongliang Z, Boya D and Lingyang S. 2019 *IEEE Transactions on Wireless Communications (IEEE)* vol. 19, no. 3, pp 1346-1359

[11] PX4 Autopilot Project, https://pixhawk.org/start

[12] QGroundControl, http://qgroundcontrol.com/

[13] FlightGear Flight Simulator, http://www.ightgear.org/

[14] Simple multirotor simulator with MAVLink protocol support, https://github.com/DrTon/jMAVSim