ABSTRACT

Unmanned Aerial Vehicle (UAV) is an aircraft with no pilot on board [1]. UAVs can be remote controlled aircraft (e.g. flown by a pilot around at a ground control station) or can fly autonomously based on pre-programmed flight plans or more complex dynamic automation systems [2]. This research focuses more on autonomous flight controlled by more flexible and smarter navigation system to search a dense forest environment. This was achieved through the use of 3D simulations, Robot Operation System (ROS) and Simultaneous Localization and Mapping (SLAM) which required high level programming knowledge in C/C++. What we have gained from this research project is the knowledge to simulate autonomous UAV search and rescue using a game engine.

INTRODUCTION

UAV simulation system plays an important role in UAV search and tasking framework [3]. Before flight tests, developed UAV hybrid system, including control, sensing, and algorithms must be tested in simulations [4]. In our group, we are developing a high fidelity 3D simulation system based a game engine. Robot operation system is used to simulation real-time control and algorithms.

BUILDING A UAV

Key Components of an Unmanned Rotorcraft System [5]
1. Radio-controlled (RC) rotorcraft
2. Avionic system for collecting inflight data, performing automatic control laws, executing mission-oriented tasks, and communicating with the ground station
3. Manual control system consisting of a pilot and a wireless joystick
4. Ground station system for monitoring the flight states of the UAV and communicating with the avionic system
A reliable unmanned aerial platform is essential as it is the foundation of subsequent work e.g. flight dynamics modelling, control system design and autonomy [6], therefore we cannot be careless when constructing the platform.

**Choosing a Platform**

There is a wide selection of platforms for us to choose from as shown in Figure 2.
Though there are many designs available, most UAVs are made up of the following:

Hardware System:
1. Quad-rotor
2. Sensor
   a. Inertial Measurement Unit (IMU) + Global Positioning System (GPS)
   b. Monocular camera
   c. 2 processor solution: flight control + vision processing

![Figure 3: Essential Hardware Components of Avionics](image)

Software System:
1. Flight control system: gumstix processor to realize real-time, measurement reading, flight control, servo driving and wireless communication [7]
2. Vision guidance system: mastermind processor to realize camera image capture, intensive image processing [8]

Navigating an Aircraft

In order for the aircraft to move in the correct flight path, physics comes into place. Figure 4 shows how physics helps.
To get navigation information, we can rely on the different forms of navigation e.g. landmarks which are recognized by human pilots, celestial navigation, radio navigation (ground-based and satellite), visual navigation and inertial navigation (gimbaled and strapdown). The inertial navigation system is a group of sensors and computing devices that observes the position, velocity, acceleration, orientation, angular rate of the vehicles with respect to an inertial frame of reference [9].

**Figure 4: Flight Description and Needed Information**

**IMU**

As mentioned above, there are 2 types of IMUs, namely gimbaled IMU and strapdown IMU.
1. Microelectromechanical System (MEMS)-based strapdown IMU [10]

Figure 6: Inertial Measurement Unit (IMU)

Figure 7: MEMS-based Strapdown IMU

Figure 8: Basic Idea for Strapdown IMU
2. Gimbaled IMU

*Building the UAV*

Materials List:
1. White Sheep 480 PCB
2. Skyline 32 Flight Controller
3. XRotor-20A-V1 (x4)
4. Brushless Motor (x4)
5. Lipo Battery
6. Slow Flyer Propeller (x4)
7. Propeller Adapter (x4)

![Figure 9: Our UAV (without the propellers)](image)

### 3D SIMULATION – UAV AUTONOMOUS SEARCH

*C# (Programming Language)*
There are many programming languages being used by programmers currently as shown is Figure 10.

![Figure 10: Programming Languages](image)
C# is a multi-paradigm programming language encompassing strong typing, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. C# is one of the programming languages designed for the Common Language Infrastructure. Our team is required to learn the C# language because the game engine that we are working on is based on C#.

**Unity3D**

Unity is a cross-platform game engine developed by Unity Technologies and used to develop video games for PS, consoles, mobile devices and websites. For this research project, we have chosen Unity as our game engine due to the following reasons:
1. **C# language.** C# is a high-level programming language which allows developers to enter the game development process easily. It is important because unlike other game engines based on C++, C# has many elements and techniques that have already been introduced. All that remains is for developers to use them.
2. **Unity is cross-platform.** This means that the same code, developed via Unity engine, can be ported on many platforms e.g. PC, Mac, Android, iOS, Web and game consoles with minimal modifications. It significantly helps reduce the effort required for the development of the game.
3. **Good community.** A good community is vital to the game development which means that each engine’s function has a clear description with example on the developer’s website. If an aspect remains unclear, the support team will answer any questions.
4. **Unity’s Asset Store, where many useful plugins and assets for game development are collected.** There are both free and paid tools, which can be found in the Asset Store through a convenient search function. Once the developer finds the tool he or she is looking for, it can be instantly downloaded and integrated.

**ROS**

Robot Operating System (ROS) is a collecting of software frameworks for robot software development, providing operating system-like functionality on a heterogeneous computer cluster [11]. ROS provides standard operating system services e.g. hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor, control, state, planning, actuator and other messages. Despite the important of reactivity and low latency in robot control, ROS itself is not a Realtime OS, though it is possible to integrate ROS with realtime code. ROS is essential in controlling how the robot behaves.
SLAM

In robotic mapping, simultaneous localization and mapping (SLAM) is the computational problem of constructing or updating a map of an unknown environment while simultaneously keeping track of an agent’s location within it. SLAM algorithms are tailored to the available resources, hence not aimed at perfection, but at operational compliance [12]. Published approaches are employed in self-driving cars, UAVs, planetary rovers, newly emerging domestic robots and even inside the human body [13]. By implementing SLAM on the UAV, it is able to use landmarks to improve the raw odometer to get a better idea of where it is and also map the area at the same time while the UAV is travelling.
RESULTS AND DISCUSSION

We were able to simulate the flight path of an autonomous UAV navigating through a dense forest map using Unity3D.

![Figure 13: Simulating flight path](image)

CONCLUSION

Through this research, we have learnt how to simulate autonomous UAV search and rescue using a game engine, and also the system structure of UAV simulation system and how we can simulate UAV tasking using game engine and robot operating system (ROS). We were able to deliver a basic 3D UAV simulation system structure and develop simple algorithms to control the UAVs.

AREAS OF FURTHER RESEARCH

We can work to improve the algorithms to make it more thorough and detailed. We could also develop more algorithms to help with the navigation of not only UAVs but other vehicles.
REFERENCES


