AmadorUAVs Macron Technical Design Paper

Amador Valley Unmanned Aerial Vehicles Team

2019-2020 AUVSI SUAS

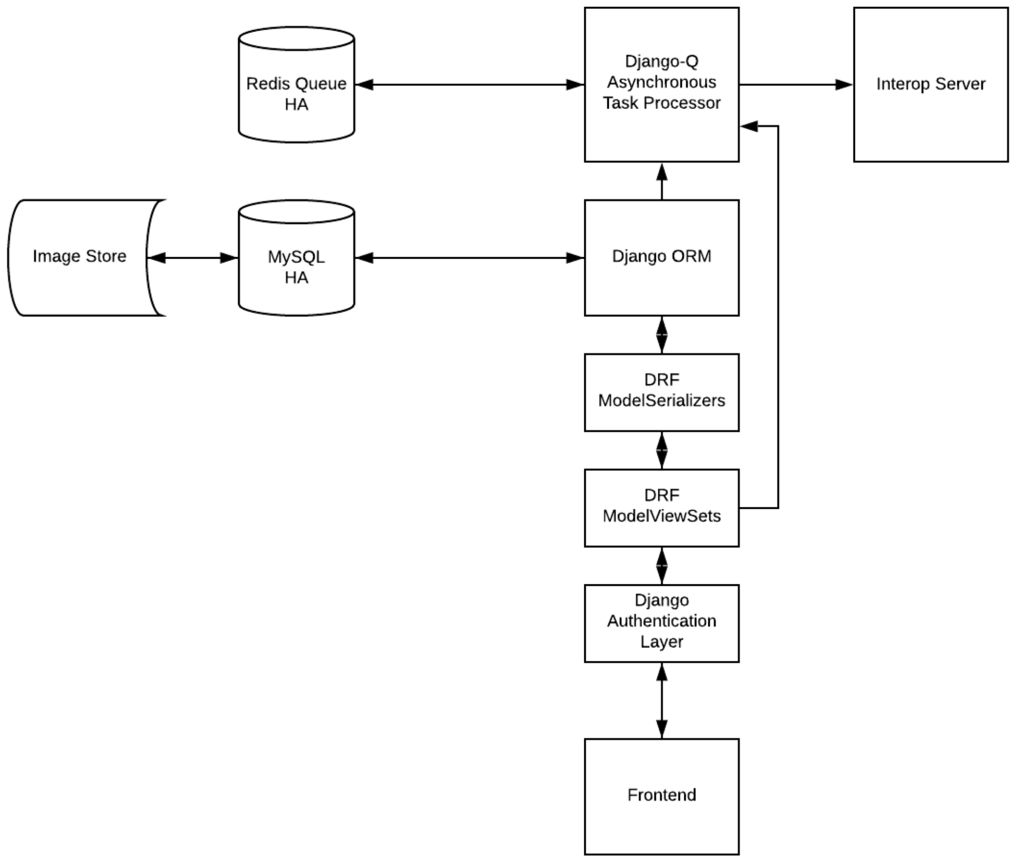
Abstract

This paper describes the technical approach that Amador Valley High School's Unmanned Aerial Vehicles Team (AmadorUAVs) took while designing their Macron UAV. As a first time competitor, the team had to draft and fabricate parts to adapt an existing multirotor frame in a manner that successfully and efficiently completed all required mission tasks. Additionally, the team had to develop a software stack from scratch that allows AmadorUAVs' complete UAS system to connect and interact with the competition's software system. The team had to carefully balance design effectiveness and quality while maintaining a strict manufacturing budget. Macron is AmadorUAVs' 2020 competition UAV and is designed to meet all competition objectives effectively.

Contents

1 Payload Design

1.1 Imaging System



Our state-of-the-art Image Management System is capable of handling multiple users and large quantities of images. The Image Management System mainly consists of a web-based frontend and a Django (DRF) backend backed by a MySQL database, and the Django-Q asynchronous task processor, backed by a highly-available Redis queue. For the competition, the Image Management System is configured as such:

1. The system presents a user with an image pulled from our drone.
2. The user creates annotations and attaches metadata.
3. The user submits the annotations; the system stores these annotations.
4. The system presents the annotations to another user for verification; once the other user verifies/updates the annotations, the system stores and accepts them, performing the necessary operations (cropping, latitude/longitude calculation, etc.)
5. The system then uploads the verified annotations to the Interop Server for judging. The annotations can be reloaded, edited, and re-uploaded at any time.

However, the Image Management System can also easily be reconfigured to serve as a simple multi-user image annotation tool, utilize a machine-learning algorithm to aid in ODLC classification or other uses. When choosing frameworks and libraries to build the Image Management System, our team had several priorities in mind:

- **Maintainability** - We wanted to create a system that could be maintained, debugged and scaled easily.

don't u
"wante
tem ove
again"

- **Readability** - We wanted our system to have an easy-to-approach code-base so that even relatively inexperienced members of our team could work on the system.
- **Scalability** - We wanted our system to be scalable to serve high image throughput and large amounts of users.

Using the Python-based Django framework (combined with Django Rest Framework to abstract API routing and handling and Django-Q to provide asynchronous task processing) proved to meet all three criteria:

too mu

- **Maintainability** - Django Rest Framework provides generic viewsets and serializers that can be easily extended.
- **Readability** - The Django ORM abstracts all database operations into an intuitive object-oriented interface, and Django Rest Framework abstracts API request handling into simple, concise, easy-to-understand view sets and serializers.
- **Scalability** - By combining Django with an asynchronous scalable task processor such as Django-Q, the team has enabled the back end to reliably handle large numbers of users classifying images simultaneously.

Our Image Management System, therefore, meets our criteria of Maintainability, Readability, and Scalability, while also doing an effective job of providing image classification and meeting the requirements of the competition.

1.2 Object Detection, Classification, Localization

Due to the technical and time overhead in designing an autonomous algorithm for classification, the team is not attempting fully autonomous ODLC this year. Instead, all classification will be done manually on the ground. The team designed an Image Management System, which consists of a Python Django web server that saves images to a Mysql database as they are gradual. streamed from the drone. TODO

1.3 Communications

Multiple radio communication technologies are employed between the UAV, GCS, and UGV:

1. A FrSky X8R long-range receiver is bound to a handheld FrSky Taranis X9D-Plus radio controller for manual and emergency control. This allows for manual override of Macron control in the unlikely event of an emergency or systems failure. The manual radio connection utilizes frequency hopping ACCST technology at a frequency of 2.4GHz using the inverted S.BUS protocol, resulting in a very reliable and secure connection.
2. For autonomous control of Macron by the GCS in addition to telemetry streaming, an RFD900x long-range telemetry radio is used. This radio has a maximum theoretical range of 80 kilometers, well above the required distance during mission execution. The telemetry radio also features hardware-accelerated AES encryption, ensuring a secure connection between the GCS and Macron.
3. For the ODLC task, a telemetry radio proved inadequate as it has a maximum bandwidth of 500kbps. A high bandwidth connection was required to transmit images from Macron to the GCS. For this, the team employed a Ubiquiti radio system. This consists of a Ubiquiti LightBeam directional antenna wired up to the server, and a Bullet Antenna connected to the JetsonNano on Macron. This allows the software stack to transmit WPA2 AES encrypted data over a 5GHZ WiFi stream at relatively high speeds.

1.4 Air Drop

TODO
winch r
writing

Winch
diagram

Softwar

2 Autonomous Flight Design

2.1 Aircraft

The team uses a modified CineStar 8 heavy-lift octocopter as the base airframe. A set of eight T-Motor MN4012 480kV brushless motors are used, each connected to a T-Motor F35A Brushless ESC.

Aircraft Length	48in
Aircraft Width	48in
Aircraft Height	18in
Empty Weight	
All Up Weight (AUW)	14.6lb
Aircraft	18in

2.2 Autopilot

Macron's autopilot is a CUAV V5+ board. This modular flight controller supported all functions required for Macron while providing IMU and power redundancy, vibration dampening capabilities, and a powerful processor. After much research and consideration, the team installed the PX4 Pro professional autopilot, part of the DroneCode autonomous drone software stack. The team chose PX4 over the more traditional ArduPilot for a variety of reasons:

Table s
ferences
and Ar

- More modular architecture, allowing easy customization and adaptation for our use case.
- QGroundControl, the standard ground station software for the DroneCode stack, is far more powerful and especially easy to use than the ArduPilot alternative, Mission Planner. It also runs on a variety of operating systems including Android and iOS systems, as compared to Mission Planner, which only runs on Windows.
- PX4 is much more actively developed, with new features coming out all the time. PX4 also has a huge international user base of professional drone testers and contributors, making it a solid autopilot.
- While AmadorUAVs strives to open source the majority of our code, PX4 has a more permissive license (BSD 3-clause) than Ardupilot (GPLv3), giving more flexibility to the adopter.

PX4 is controlled using the MAVLink (Micro Air Vehicle Link) communication protocol. The team developed a modular navigation system, called Pilot, to implement all autonomous mission objectives. It is packaged via Docker containers and deployed onto our ground station, onto our in-house Kubernetes cluster. Kubernetes is an industry-standard container orchestration system, which the team uses as a host for all ground-based software (Pilot, IMS, etc). Running on top of a Dell EMC R410 server, the system has very high reliability.

Add Pi
Stack d

change

2.3 Obstacle Avoidance

The team implemented 2 types of obstacle avoidance: the first is to implement the Virtual Obstacle Avoidance mission requirement; the other is to prevent physical collisions with other competition UAVs and other environmental obstacles.

2.3.1 Virtual Obstacle Avoidance

The team integrated a virtual obstacle avoidance solver into the Pilot autonomous navigation system. The solver is as follows:

TODO
and ima
mation
code

1. The UAV's flight path is loaded into a graph structure.
2. Virtual obstacles are requested from the Interop Server and loaded into the solver.
3. The solver draws a 3D "mesh" of points around the cylindrical obstacles (with a 1.5x margin) and loads them into the graph structure.
4. Vertices that can be traversed by the drone are connected, and weights are calculated based on flight difficulty (flight time, distance, sharpness of turn).
5. An A* algorithm is run over the dataset, producing an optimal path with minimal detours while still avoiding obstacles.
6. This path is exported into a flight plan format, and this modified flight plan is uploaded onto the UAV.

2.3.2 Physical Obstacle Avoidance

While less of a concern, the team took measures to implement physical obstacle avoidance against other aircraft and environmental obstacles. Due to the tight budget of AmadorUAVs, the team was unable to afford a LiDAR systemfor real-time obstacle avoidance of moving obstacles. However, the team took advantage of the extended telemetry capabilities of the 2020 Interop Server: the ability to query other flying teams' telemetry data. By constantly polling for updates on the position of other UAVs in the competition, the team's GCS system plots out a predicted trajectory of each aircraft in real-time, then conveys this data to Macron. Macron's onboard JetsonNano computer uses this data to provide obstacle avoidance capabilities to the autopilot. This functionality makes use of PX4's built-in obstacle avoidance/trajectory planning protocol, which allows the onboard computer (JetsonNano) to modify the trajectory plotted out by PX4 in real-time via the MAVLink protocol. This functionality allows the onboard computer to verify that the plotted trajectory does not collide with any known obstacles, and make modifications as needed.

LiDAR

3 Safety, Risks, & Mitigations

As safety is of utmost importance during the development of the UAS, the team prioritized safety during the development process and mission tests.

Write t

cliche a
plate, c

3.1 Developmental Risks & Mitigations

During the development of

Risks d
velopm

3.2 Mission Risks & Mitigations

Risks d
comps,
etc

Referen