Redbird Robotics Technical Paper for IARC Mission 9

ABSTRACT

Redbird Robotics of the University of Louisville has designed an autonomous vertical take-off and landing (VTOL) aircraft with the aim of successfully completing the tasks presented through IARC Mission 9. The vehicle integrates custom flight hardware with a software stack comprised of Robot Operating System (ROS) and PX4.

INTRODUCTION

Statement of the Problem

IARC Mission 9 presents participants with the challenge of successfully navigating to, manipulating, and replacing a target module under a set of various design and environmental constraints. The target is attached to a moving mast that the aircraft must be capable of recognizing and tracking to achieve this goal. In addition, the aircraft must travel around the specified course to reach the target under a time constraint, making cruise speed an important constraint.

Conceptual Solution to Solve the Problem

Figure of Overall System Architecture

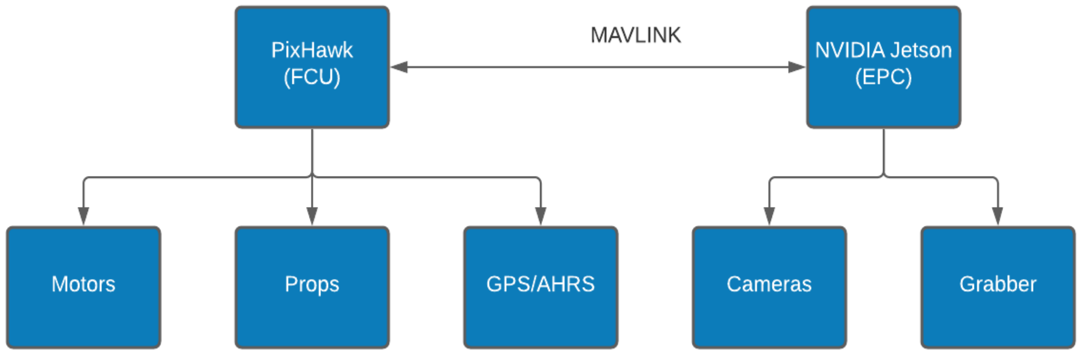

Figure 1. *Figure of Overall System Architecture*

Yearly **Milestones**

- February 2021 – Simulation environment implemented
- June 2021 – Implemented simulated depth sensing
- June 2021 – Complete aerodynamic design
- July 2021 – Develop carbon fiber manufacturing skills

- July 2021 – Pose estimation and object classification utilizing object detection software
- October 2021 – Implement autonomous navigation (projected)

AIR VEHICLE

Description of Configuration/Type

The vehicle is a vertical take-off and landing (VTOL) aircraft with one pusher-propeller and four vertical propellers for hovering. The plane has a 1.5 meter high-mounted wing. This configuration allows for conventional quadcopter-style vertical takeoff and conventional plane-style forward flight. Plane-style forward flight will improve range and efficiency at the cost of negligibly more weight.

Flight Control System

Navigation/State Estimation System

The navigation system is based on the standard ROS 2D navigation stack, with modifications added to adapt to an aerial environment. The drone is programmed to achieve a set altitude upon takeoff, at which point it constrains its movements to a 2D plane parallel to the ground plane at that altitude. As the drone begins each run in a known position relative to the competition field and the field itself is static, a global plan from the drone's initial location to its goal position need not be dynamically generated. This eliminates the need for a global planner and global map. Dynamic obstacles are accounted for by using a 2D costmap representation of the environment, which the local planner uses to plan alternate routes around any obstacles not accounted for by the global plan. To function effectively, the system needs to be able to maintain localization and exhibit resistance to error in its odometry sources. Currently the system relies on fusion of high-accuracy GPS data with readings from the onboard IMU to maintain an accurate estimation of the system's current position on the field.

Attitude/Position Control System

The attitude/position controls are abstracted from the primary software package via the MAVROS API. This is a ROS interface that allows the primary application to communicate with the flight-controller unit (FCU) via the MAVLINK protocol. The FCU is loaded with the PX4 flight-controller firmware, which is capable of handling low-level flight functions such as RC, position/velocity control and limited autonomous navigation. Given a position or velocity setpoint, the firmware can drive the motors to achieve the state specified by the setpoints.

Figure of Control System Architecture

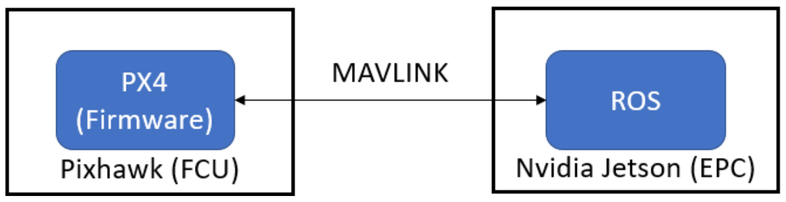

Figure 2. The object detection software architecture

Flight Termination System

MISSION PACKAGE

Perception System

Target Identification and Behavior

The target identification functions of the vehicle are implemented through a multitude of discrete hardware and software components that work together to transform raw sensor data into useful descriptions of the environment state. The general object detection and identification architecture can be represented as follows:

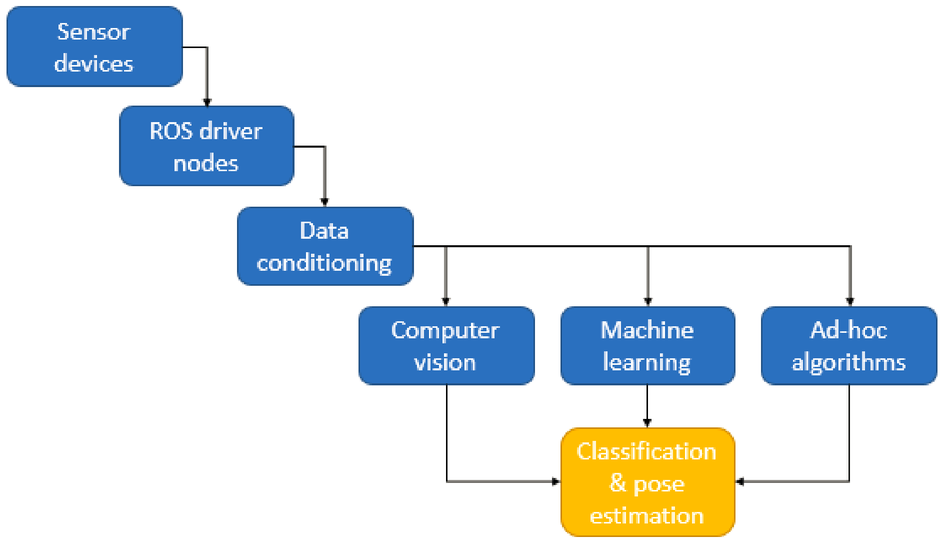

Figure 3. The object detection software architecture

Each node represents a different stage in the object detection and identification process. In the first stage, RGB camera and depth camera sensors record raw data from the surrounding environment. This data is then made available to the rest of the software stack through ROS packages that provide APIs to access camera data. This data is then validated, and afterwards transformed through various processes to remove noise and otherwise provide a more useful and accurate base for the following object detection stages. The conditioned data is then passed along to be processed by the object detection stages, which are comprised of separate procedures that work in parallel to independently detect regions of interest in the input data. The final step involves using the output of these different algorithms to classify the object and estimate its position in space.

The object detection code is primarily based upon the OpenCV software library and uses a variety of utilities provided by it throughout the process of detection and identification. The image processing tools provided by the library are used to perform tasks such as image conditioning and color filtering. The machine learning modules are also used to aid in processes such as text detection and feature matching between frames. Other ad-hoc algorithms outside of these are also used to perform such tasks as statistical analysis on depth point data.

Threat Identification and Behavior

There are two main ways the system identifies and responds to threats/obstacles. The system perceives its environment using Intel Realsense D435 cameras, which are used to generate 3D point-cloud data representing the drone's surroundings. When path planning, obstacles that appear within this point cloud are projected onto a 2D occupancy grid, which is used by the local planner to plan paths around any dynamic obstacles not accounted for by the global plan.

The system has a redundant layer of safety to prevent collision, known as the "safety-velocity" system. This system acts independently of the planning software, and specifically monitors the proximity of obstacles to the system. The system uses the concept of multiple "proximity regions", which can be represented geometrically as nested spheres of varying radii each with origins about the drone. If an obstacle enters one of these regions, the drone will respond by reducing its velocity in correlation with the obstacle's distance from the drone. If an obstacle were to enter a region that corresponds to a position in space that is too close for the system to take evasive action, the safety-velocity system will trigger an e-stop event which will cause the system to cut power to all propellers and fall to the ground.

Gesture Identification and Behavior

To match the unpredictable movements of the mast to allow for precise manipulation of the communications module, more on-board software is necessary. One of the primary techniques used involves a series of steps relating to the detection, filtering, and matching of key features detected in different frames. After these features are matched between frames provided by the sensor cameras, the total transformation of the mast pose can be calculated, and from there necessary adjustments can be made to the vehicle's flight pattern to maintain a constant relative position to the mast.

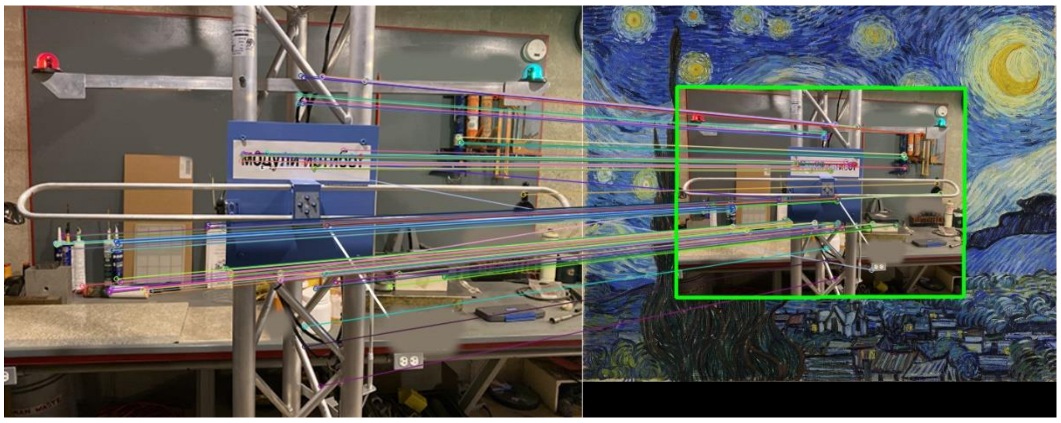

Figure 4. The green border was generated by the detection algorithm after matching common features between the original and transformed (scaled) image.

On the initial trip, once the pose of the vehicle relative to the mast is within a certain spatial region, the grabber mechanism is activated, and the communication module is removed from the mast. The return trip is very similar, apart from attaching the replacement module in a similar fashion.

Communications System(s)
Communication happens over multiple wireless links. Video, flight data, ROS diagnostic data and other important information all must be communicated. To do this a few different antennas and frequencies are used. These frequencies and protocols are detailed below.

Frequency	Protocol	Purpose
915MHz	MAVLINK	Telemetry
5.8GHz	PAL	Video for debugging
2.4GHz	802.11	ROS Debugging

Table 1. An outline of communication protocols used by the vehicle.

User Interface / Man-Machine Interface
For debugging purposes, the development team can command/interact with the system using multiple interfaces. These include a shell interface via the remote-shell utility ssh, data visualization and control using the ROS tool rviz, and standard RC joystick input. During the competition, none of these interfaces are required, as the system is required to be completely autonomous. Therefore, the only communication link that will be online during a competition run is the required kill-switch to signal the drone to trigger an emergency-stop event.

RISK REDUCTION

Vehicle Design

Our vehicle is designed with safety in mind, several design decisions contribute to the safety of our vehicle. Firstly, lightweight materials have been selected so that the momentum of the vehicle is not excessively high. Secondly, in steady state fixed wing cruise our drone is propelled by a rear facing pusher-propeller that is more safely located than in puller-propeller configurations. Thirdly, our wing has been optimized to be able to fly at relatively low speed. This gives the drone more decision-making time, and information gathering time and keeps the energy of the system low.

EMI/RFI Solutions

Our vehicle makes use of carbon fiber. Carbon fiber is an effective attenuator of EMI. Grounded carbon fiber can allow us to shield internal electronics from unwanted outside noise.

Shock/Vibration Solutions

Electronics and other vibration sensitive electronics are to be mounted on top of rubber gromets so that in the case of shock or vibration they are largely projected.

SOLUTIONS

Safety

The prototype is designed with redundant failsafes to present collision or an otherwise dangerous malfunction. The navigation-stack/path-planner is responsible for actively directing the drone to the most efficient path towards its destination that does not intersect with any perceived obstacles. There is an independent velocity control system that regulates velocity based on the drone's proximity to the nearest obstacle, triggering an e-stop event if an obstacle gets too close. Lastly there is the manual killswitch, which can be triggered by a manual operator in order to interrupt power to the system and cause the drone to fall to the ground.

Modeling and Simulation

The software solution for the system is capable of being simulated in software-in-the-loop (SITL) mode using the robotics/physics simulation engine Gazebo. In this mode, the software solution receives sensor and other data from simulated sources instead of the physical hardware and processes it using the same routines as if it were deployed in the physical competition environment. This allows the software team to develop features in the absence of physical hardware, which is not always readily available. Furthermore, safety-critical features that affect obstacle avoidance and navigation can be tested in the simulation environment before being deployed on the hardware platform to ensure their safety without risking physical damage to the prototype or the physical environment.

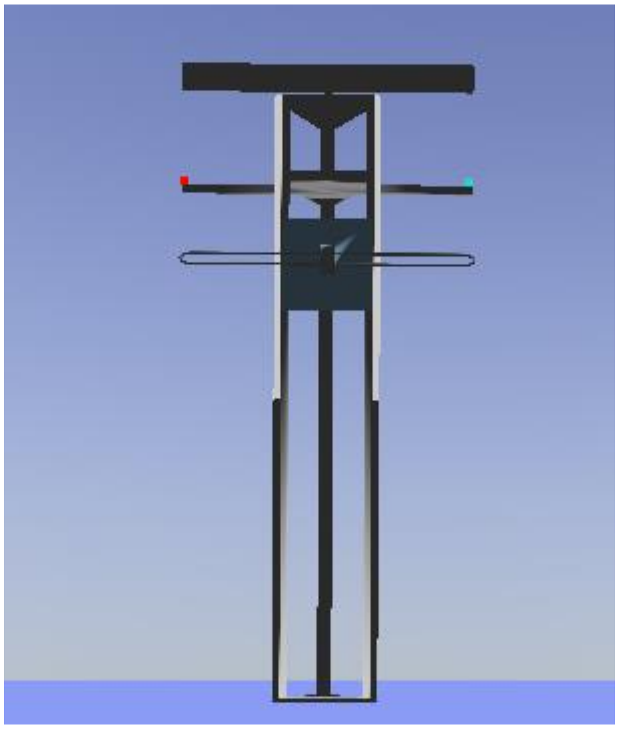

Figure 5. The simulated mast model in Gazebo as seen through the RGB camera sensor.

The design of our airfoil was modeled, optimized, and simulated using a variety of computational design tools. Aero Sandbox and XFLR5 were used for the simulation and validation of the wing. Aero Sandbox provides the unique capability to rapidly optimize wing planforms, coupled with the airfoil analysis and dynamic stability analysis tools included with XFLR5 we can be confident in our design.

Physical Testing

In addition to the tests conducted through Gazebo simulations, physical test runs are also utilized to ensure the functionality of the vehicle. The drone is tested in both a replica environment of the actual arena, as well as in a dedicated aerial drone test environment. This includes testing of safety features such as the kill-switch and obstacle avoidance systems in an actual physical environment, as well as general performance characteristics of the vehicle. We expect to be able to use indoor testing facilities with netting and motion capture for improved safety.

CONCLUSION

Redbird Robotics has designed an aerial vehicle optimized to complete the mission safely and effectively. The system uses a 2D costmap representation of the environment to plan paths around dynamic obstacles and can safely avoid collisions. The system uses computer vision and feature recognition to estimate the pose of the mast in real time, which allows the system to approach and manipulate the mast to score the goal. The system has redundant failsafes to prevent collisions including automatic proximity e-stop and manual killswitch capabilities. As such, Redbird

Robotics is confident its Mission 9 competition platform and looks forward to upcoming competition season.

REFERENCES

1. Robot Operating System https://www.ros.org/
2. MAVLink “Micro Air Vehicle Communication Protocol” https://mavlink.io/
3. PX4 “Open Source Autopilot” https://px4.io/
4. Aero Sandbox “Python package for aircraft design” https://github.com/peterdsharpe/AeroSandbox
5. XFLR5 “analysis tool for airfoils, wings and planes” http://www.xflr5.tech/xflr5.htm
6. OpenCV “real-time optimized Computer Vision” https://opencv.org/
7. International Aerial Robotics Competition http://www.aerialroboticscompetition.org/