Intelligent Quadcopter for IARC 2019

Umer Salman, Noel Brownback, Eric Johnson, Mario Gonzalez
The University of Texas at Austin

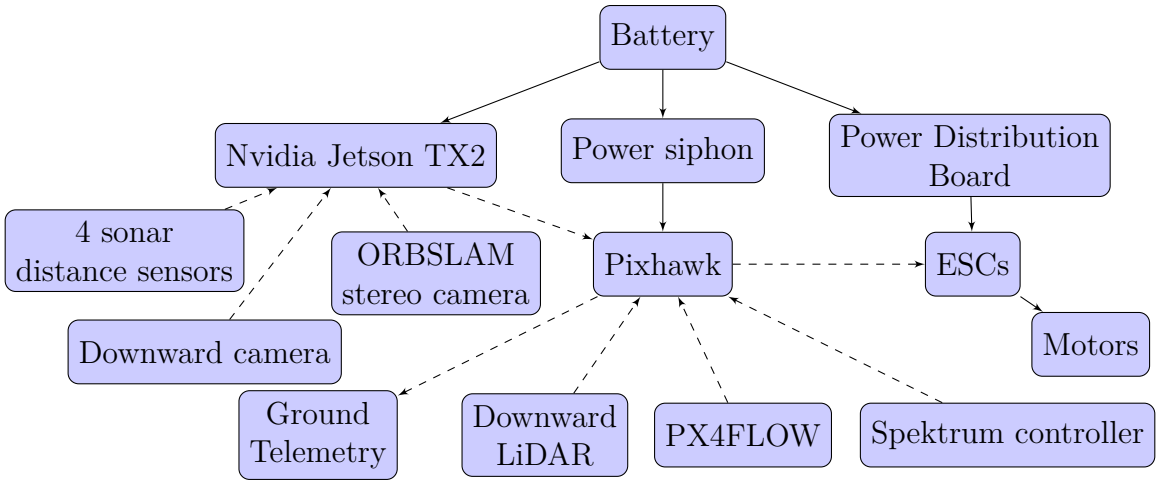
ABSTRACT

Texas Aerial Robotics has constructed a custom quadrotor aircraft system for the fulfillment of the International Aerial Robotics Competition Mission 8. A combination of lightweight materials and powerful propulsion optimizes flight time, while standardized flight control systems supported by a massive community enables customization and adaptation to the mission at hand. This includes the integration of a multitude of sensors that enhance environmental awareness and allow for the aiding of people to analyze and interact with an environment. This is done through homebrewed QR code decoding algorithms and the shared computing power onboard the multiple aircrafts. Safe development has been assured by strategic selection of components, thorough vetting of software in Software in the Loop simulation, and robust testing procedures in secure areas.

INTRODUCTION

Previous aerial robotics missions have always allowed for GPS or other external active position estimation that allowed for constant locational awareness. Location data is important to any mission, as proper state awareness prevents various failures, including complete loss of control for the drone. Thus, Mission 8 requires novel solutions to deriving this critical state data as GPS is forbidden. Additionally, the aircrafts must be capable of taking commands in a humanized "natural" fashion and avoid other aircrafts without being informed of those other aircrafts' locations.

Texas Aerial Robotics addresses these challenges by utilizing computer vision. TAR uses ORBSLAM2, an "Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras" to create maps of the arena in real time. The drone creates pointclouds of the objects in view using the stereo camera and combines sections together to create the maps. This means that should a section of the map start showing drift, the ORBSLAM2/ORBSLAMM System will toss out just the section while maintaining as much of the map as possible. TAR hopes to combine the processing between the multiple drones to have them work together on one map. These drones also solve for the QR code that is the combination the human needs for collecting the objects. TAR implements the "humanized" control aspect of the system through voice control using an Android app that connects to all of the air vehicles. The drones are also built with safety features for a plethora of other objects that may be on the field through its man-safe propellers for any humans and sonar sensors for avoiding other air vehicles at all times.



Flowchart of quadcopter electronics
**Solid arrows indicate power while dashed lines indicate power+data*

Texas Aerial Robotics Yearly Milestones

- 2017 (Mission 7) - Have a working bot, with computer vision software in nearly working state
- 2018 (Mission 7) - Continue development of computer vision, develop a computational game strategy through creation of simulations of the motion of Roombas, and tweak the physique of the drone for better flight control and stability
- 2019 (Mission 8) - Complete a working system, with localization and GPS-denied navigation of multiple drones. These drones should work together to solve the QR code and communicate with a person through an Android app that can transmit voice commands to the drones.
- 2020 (Mission 8) - Finish all aspects of mission through more networking and collaboration between the quadcopters, improve the drone for better flight control and overall size, have better communication between drones to know where enemy drones are at all times, add aesthetic design choices, and test for hours in gyms in as many conditions as possible.

AIR VEHICLE

Propulsion and Lift System

The vehicle utilizes an y axis symmetrical quad-rotor system. This H-frame provides a stable quadcopter for a smaller overall package while maintaining plenty of room for electronics. The frame is lifted by four T-Motor MN2212 KV920-V2.0 motors. These motors provide ————fix these numbers———- 1.8 [kg] of thrust per motor at 100% throttle when paired with a 4-cell battery and 9.5 [in] propellers for total 7.2 [kg] of thrust. This is necessary for our 3.15 [kg] weight to maintain a 2.2 thrust-to-weight ratio, ideal for battery life and flight time.

The propellers used with the motors are 9.5 [in] long and made of plastic. These T-Motor propellers provide excellent precision, durability, and efficiency. The rotors are placed to counter-rotate for flight stability and to retain a one-to-one ratio of propellers to motors.

Power Management System

The quadcopter uses a 10000 [mAh] 4S 10C LiPo battery and a 2000 [mAh] 3S 15C LiPo battery for power. The 10000 [mAh] 4S 10C LiPo is used to power our ESCs and Motors. With 4 T-Motor U3 K700 motors running at close to the ideal 50% throttle, a 4 Cell battery was necessary to reach the 1800 [grams] of thrust per motor needed for hovering flight. The 2000 [mAh] 3S 15C LiPo will power the flight control board and Jetson compute units.

$$T_{min} = \frac{\text{Battery power}(AH) * 60}{\text{Total current drawn by motors}}$$

Assuming an average throttle of 65% for maneuvering, we see a 9 A current draw from each motor, and with a 10 Ah battery, that provides us with 16 minutes of flight time with minimal maneuvering and height change. This gives us plenty of headroom for more intense altitude and positioning changes, which will allow us to move more quickly during the competition, a valuable asset with the given time limitation.

$$\text{Max continuous Amp draw } (A) = \text{Battery capacity } (Ah) * \text{Discharge rate } (C)$$

To save on weight and space, we chose a 10C discharge rate battery, as this is still more than enough to match the maximum continuous current draw of 25 [A] and maximum practical current draw of 19.4 [A] required by the motors. The 10000 [mAh] battery with a 10C discharge rate gives us a maximum continuous current draw of 100 [A], matching the limit of the motors.

The eCalc online tool was used to validate our calculations, and as seen by the generated graph below, there should be no issue reaching above 10 minute flight times at speeds below 15 [mph].

As for the 2000 [mAh] 3S battery, this battery powers our Pixhawk flight controller and our three Jetson TX2 compute units. This was done partly to dedicate the entirety of the main battery to propulsion to maximize flight time. The other reason is that the speed controllers we chose (T-Motor Air 40A) have no BEC, so a separate battery for the flight controller (and in turn the receiver) was the simplest solution. 2000 [mAh] was chosen because of the slim form factor we were able to find it in, and because it will provide more than adequate power while not adding any significant amount to the quad weight.

Flight Control System

Navigation/State Estimation System

Our system builds off of the robust open source project ArduCopter. ArduCopter, which runs on the Pixhawk, controls the stability and position control of the drone by taking in

data from the IMU, compass, altitude ToF, and a position estimate generated from our optical flow sensor or ORB SLAM. The ArduCopter flight stack is maintained by hundreds of developers from around the world and the software is deployed on thousands of commercial and recreational drones. The vast community of developers and users ensures reliable controls code with lots of features.

We have made slight modifications to the ArduCopter firmware, which allow us to set the EKF origin from our GNC ROS node. This has been done to make our drones capable of navigating in GPS denied environments. For general waypoint navigation, the drones utilize the ArduCopter EKF that takes in data from the sensors listed above, including the optical flow, compass, and IMU. The EKF position is then queried via mavros so that our navigation code can use our position to make intelligent decisions. TAR has greatly improved ArduCopter's GPS denied capabilities by integrating ORBSLAM into the ArduCopter EKF. The use of ORBSLAM allows the drone to navigate based on 3 dimensional visual features in the drones operating environment. ORBSLAM continuously builds a map of its environment and compares current local maps to its master. Because of this comparison algorithm, ORBSLAM is able to perform loop closure which is useful in negating any propagated error from the rate sensors.

In addition to the stability and position control from ArduCopter, TAR's drone sports a navigation node, which executes our pathplaning algorithms developed for Mission 8. The navigation node determines a waypoint and mode, then does a sanity check on the waypoint before publishing the waypoint to ArduCopter until the drone has reached its destination.

Attitude/Position Control System

A crucial aspect of navigating in a GPS-denied is concrete information on our position. Using a Teraranger One Time of Flight sensor from Terabee, our drones are able to know their exact altitude. This sensor was chosen due to its extremely light weight over other LiDAR options. This communicates over I2C directly to the Pixhawk where ArduCopter uses that information in the EKF.

The position control aspect is handled through ORBSLAM running on the Jetson which is then fed into ArduCopter on the Pixhawk. The stereo camera in the front of the aircraft allows visual and depth information of the objects on the field. This information is used to formulate an estimate of the drone's position based on the 3 dimensional visual features of the environment. In addition, these features are compared to features the drone has already seen in order to perform loop closure. Loop closure is then used to correct for propagated error accumulated from the drone's rate sensors. No sensor or algorithm is foolproof. ORBSLAM has some limitations which limit the algorithms ability to generate position estimates. First, if the camera is moved too quickly, the deltas between frames is too large and ORBSLAM can lose tracking. This is mostly mitigated by limiting the drones flight velocity. The second limitation of ORBSLAM is the requirement of textures and 3D features. Both of these risks are low when operating in the Mission 8 arena, however, this is a flight critical system and the consequence of failure are catastrophic. TAR has opted to implement redundancy by attaching the PX4FLOW sensor to supplement ORBSLAM in the event of failure. While

not as useful on the non-textured surface, it should at least ensure that the drone does not lose control.

The quadcopter uses eight sensors for flight control. The main sensors used for the navigation of the quadcopter include a downward-facing 1D LiDAR module, a PX4FLOW optical sensor, and the built-in sensors of the Pixhawk 2 flight control board.

The 1D LiDAR module used is a Time of Flight (ToF) sensor named the Teraranger One. This one-dimensional sensor is directed downward and provides data to determine altitude. The PX4FLOW sensor measures velocity by comparing frame by frame images and measuring the distance traveled and direction. As previously stated this sensor is a redundancy for ORBSLAM. The Pixhawk 2 contains three 3-axis accelerometers, three 3-axis gyroscopes, and two 3-axis compasses. The data from these sensors is used by the Pixhawk onboard system for flight control and maneuvering of the quadcopter.

Flight Termination System

We will be using ArduCopter's EmergencyStop to act as our killswitch. In the event of catastrophic failure, the safety operator can send the kill signal, and all power will immediately be cut from the drone's ESCs, and in turn, the motors.

MISSION PACKAGE

The drone has been designed to carry all the necessary GNC sensors, an array of sonar sensors, a stereo camera, a downward camera, and a Nvidia Jetson TX2. In addition, the drone carries a telemetry radio, WiFi antennas, and an RC receiver in order to carry out safe flight.

Perception System

For completion of Mission 8, the drone must be able to detect the QR code and avoid obstacles, both moving and non-moving. The quadcopter is equipped with a single downward facing camera for reading the quadrant of the QR code. A stereo camera is used for ORBSLAMM for localization. There is also an array of sonar sensors for obstacle avoidance.

Target Identification and Behavior

For target identification and behavior, the drones fly to defined waypoints where the quadrants of the QR codes are located. Each quadcopter is equipped with a singular ELP webcam facing downward. This camera was chosen for its very light weight while still allowing adjustment of exposure and other parameters to better read a QR code on an iPad.

Once the QR code quadrant is in view, computer vision begins to process using our algorithm that uses the inherent features of QR codes. Any drone is able to narrow down the 4 digit combination to approximately 6 permutations that from only the bottom-right quadrant. The drones continue to scan the other sections of the QR code and piece together the combination. If the top-right quadrant is found, then the code is complete with only the

top-right and bottom-right sections.

After the code has been identified, the quadcopters fly to positions on the field where the human would likely choose to heal and await other commands.

Threat Identification and Behavior

Our avoidance system takes data from an array of sonar sensors. These 4 MAXBOTICS 7m I2C sonars allow the drone to detect objects on the sides the quadcopter intends to move in. This data is passed to the navigation node, which uses this data to determine if an obstacle drone or a wall is within 1.5 [m], in which case the node will issue a waypoint opposite the detections. Edge cases and corner cases have also been accounted for, such as the case where an enemy drone pushes a helper drone against a wall. In this case, the drone will either rise or fall in altitude, depending on the distance from the ground. Each drone has its own array of sonar sensors, so even in the event of communication failures between quadcopters in the full system, each drone will continue to intelligently avoid dangerous situations.

Sonar sensors were chosen for obstacle detection rather than a 2D LiDAR or webcams to save on weight, space, and computing power. Sonar sensors are very tried and true. As we use I2C as our communication protocol for these sonars, adding or subtracting sensors to the array becomes trivial, with easy addition of more sonar sensors (up to a limit of 127) with very few modifications to the code, electrical systems, and/or hardware, aside from the physical mounting of the new sonars themselves. Additionally, sonar was picked over radar as radar has a harder time picking up smaller objects, like a drone.

Other threat identification systems, such as an infrared cameras, were discussed. However, potential interference from sunlight and other environmental factors led the team to choosing sonars instead.

Gesture Identification and Behavior

Android app, PocketSphinx.

Communications Systems

On each Jetson, data is passed between software nodes via ROS. Commands to the flight computer from that Jetson are passed via serial connection with data encoded in MAVLINK messages. Data between our multiple Jetsons (1 per drone) is passed via ROS messages through a 2.4 GHz WiFi network. The quadcopters connect to Texas Aerial Robotics' router, which maps the quadcopters to IP addresses. The drones are interchangeable on the network-side too as the drones to IP addresses is not hard defined, resulting in a more modular and more robust system.

Man/Machine Interface

The drone has a DX7 RC controller interface for system checks and emergency situations. In addition, the flight controller can be monitored via a telemetry radio using Mission Planner.

Our autonomous scripts must be initiated by an SSH connection to our Nvidia Jetsons.

OPERATIONS

Flight Preparations

Checklist

1. Check battery voltage (12.0-12.6V) and insert
2. Make sure LiDAR, PX4FLOW, and cameras are not covered
3. Make sure that the props are on in the correct direction (leading edge in)
4. Make sure prop guards are secure
5. Make sure Kill Switch is in correct position
6. Power on controllers
7. Make sure ground station has good telemetry
8. Verify critical sensors are giving good data
9. Plug in TELEM2
10. SSH start/verify autonomous scripts are running
11. Make sure everyone is clear of drone
12. Verify mode switch in Stabilize
13. Hold safety button
14. Mode switch to Autonomous

RISK REDUCTION

Vehicle Design

EMI/RFI Solutions

The Arducopter flight stack helps keep interference risk down through the EMI motor calibration on the Pixhawk. The calibration calculates the magnetic interference correction by formulating a function of the magnetic interference based on the power setting of the motors. Additionally, the drone was designed to reduce the electromagnetic interference in the first place through the use of carbon fiber over the high voltage electronics. Antennas and other radio connections are then done above the carbon fiber.

Shock/Vibration Isolation

The modern Pixhawk flight controller contains internal damping from the vibration of the drone's structure. The team decided that there is no need for extra damping. After completion of our drones, flight logs from the Pixhawk confirmed that vibration on the quadcopter was well within recommended amounts. The team believes there is no need to have landing shocks, as the sophisticated ArduCopter flight stack has logic to recognize landing. Because of this, nominal landings will not interfere with the guidance or navigation of the drone.

Safety

We took significant steps to ensure safe operation of the quadcopter. In order to keep the quadcopter from flying where we do not want, we have a Spektrum receiver module onboard

so our designated pilot can manually control the vehicle. Additionally, we are able to check the status of the quadcopter through our telemetry to our ground station. The structure of the drone is also outfitted with prop guards to mitigate damage to people or property in the event of catastrophic failure. Our prop guards took multiple iterations. We wanted a design that would not interfere with the prop wash, but still be able to prevent damage to the quadcopter itself in the unfortunate event of a collision. The first iteration looked nice, but interfered with the prop wash too much and weighed too much, reducing the thrust. However, the next iteration cut weight and did not interfere with the prop wash.

Modeling and Simulation

To safely test our software throughout the development process, the team implemented a simulation using the Gazebo application. Using this tool, TAR was able to deploy full-scale software in the loop simulation without jeopardizing hardware with each design iteration.

To employ the Gazebo framework, the team imported a standard quadcopter model from the ArduCopter standard library and Roomba models from Gazebo model library, which were then updated with the color-coded plates and obstacle tubes prescribed for the competition. Additionally, the ground plane within the reproduction was changed to match the texture expected at the competition.

Gazebo was chosen because of its compatibility with ROS and Ardupilot, the internal communication protocol employed onboard our drone. With this congruence, the exact same software run onboard the drone could be run within the simulation, complete with sensor feedback detected within the simulated instance. C++ scripts could then be utilized to enforce the correct Roomba behaviors, scripting the robots using the same messaging environment as the quadcopter itself (ROS).

Overall, modeling our software behavior within a Gazebo simulation greatly expedited the development process. By standardizing the simulation setup, tests could be conducted on multiple computers at any time, instead of requiring the sluggish process of updating, calibrating, and flying the physical drone. Additionally, simulation protected the hardware from the accidents and unintentional flight behavior inherent to prototype software.

Physical Testing

Simulation, however, can only prove so much. Eventually, refined software was tested on a model testbed or on the drone itself. Much of the computer vision and general software integration was developed on this testbed, where performance could be verified with the same sensors and computational hardware as the aircraft before full-scale deployment. Even strategy nodes could be tested on this platform, with waypoint predictions vetted before being deployed to the aircraft.

Flight tests, expectedly, require a specific setting. Generally, Texas Aerial Robotics tested on the roof of the Aerospace building on campus at The University of Texas at Austin. This controlled environment bore witness to hardware implementation throughout our design

process, from original flight readiness to early autonomy.

However, summer development saw additional test flights conducted in a vacant parking garage, but both locations suffered from magnetic inconsistencies which interfered with compass calibrations. Because of this, later tests, including the qualifying run submitted, were conducted in a parking lot to the north of the university.

Regardless of the setting, each flight test was conducted with constant apparatus: a four meter by four meter test mat printed with the pattern expected at competition to allow optical flow for navigation. Across this mat drives a test Roomba, which accurately recreates both the physical appearance and behaviors of those robots that will be tracked at the venue, complete with top plate and switch. Finally, there is the drone itself, which starts on a corner of the mat, takes off autonomously, and then executes the maneuver to be tested.

CONCLUSION

Texas Aerial Robotics' solutions to the challenges provided by IARC Mission 7 are bountiful and diverse. The physical design process, from the ground up, optimizes flight time while enabling the physical contact required for success. Unique software utilizes input from an expansive collection of sensors to overcome the telemetry limitations imparted by the contest and track the dynamic objectives as they move about the area. Safe operation of these vast networks of interlocking components has been ensured through particular hardware selection and thorough vetting.

References

[1] HA Daoud, Sabri AQ Md, CK Loo, and AM Mansoor. Slamm: Visual monocular slam with continuous mapping using multiple maps. *PloS one*, 13(4):e0195878, 2018.

[2] Raul Mur-Artal and Juan Tardos. Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras. *arXiv preprint arXiv:1610.06475v2*, 2017.