Autonomous Herding of Ground Robots Using Computer Vision State Estimation and Tracking with an Aerial Robot

Kevin Sheridan
Purdue AUVSI - Purdue University
Logesh Roshan Ramadoss
Purdue AUVSI - Purdue University
Jason King Ching Lo
Purdue AUVSI - Purdue University

ABSTRACT

In this paper, we describe our system for Mission 7 of the International Aerial Robotics Competition. The goal of Mission 7 is to autonomously herd 10 ground robots by touch over one of the four edges of a 20 meter by 20 meter arena in a GPS denied environment while avoiding moving obstacles. Our solution uses a neural network to determine which ground robot to touch and how many times to touch it. We use a Mini-ITX motherboard with an Intel I5 and ROS as our main computer onboard the robot. Our robot has 5 cameras, 1 LiDAR, and 1 IMU. We localize by fusing visual odometry, a particle filter, and IMU readings in an EKF. We use a minimum time polynomial trajectory generator to feasible paths for execution with a closed loop controller.

INTRODUCTION

Statement of The Problem

Mission 7 of the International Aerial Robotics Competition, IARC, is to autonomously herd 10 autonomous ground robots over one side of a 20 by 20 meter square arena. In addition to the 10 ground robots, there are 4 obstacle robots with cylinders of up to 2 meters high. All four of the obstacles move on a circular path stopping only when they collide with an object. When touched, the ground robots turn 45 degrees. When blocked, the ground robots turn 180 degrees. Every 5 seconds, the ground robots rotate -20 to 20 degrees randomly, and every 20 seconds, the ground robots turn 180 degrees. The aerial robot may not use GPS to localize.

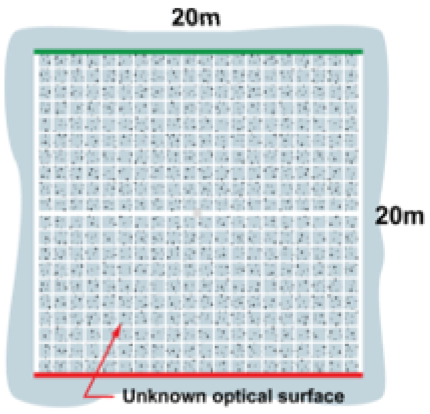

Figure 1. Arena

Conceptual Solution to The Problem

We decided to use a large quadrotor. We chose this aircraft because of the simplicity of its dynamic model and mechanics.

Our sensors consist of 5 color cameras mounted on all 4 sides and the bottom of the quadrotor, a LiDAR, and an IMU. We localize using solely computer vision, and track the ground robots using computer vision. Due to the enormous amounts of data streaming from all 5 cameras, we decided to use a full Mini-ITX motherboard with a quad-core Intel i5 cpu. This is one of the few reasons we chose to use a large quadrotor.

To interact with the ground robots, we use a 6 degree of freedom robot arm mounted on the bottom of our quadrotor to physically tap the ground robots. This is different from most previous approaches, which consisted of landing on the ground robot to interact with it.

We used machine learning to teach a neural network to choose which ground robot and how many times to touch it. We also employ a minimum time polynomial trajectory generator to compute efficient and agile trajectories to the target robot.

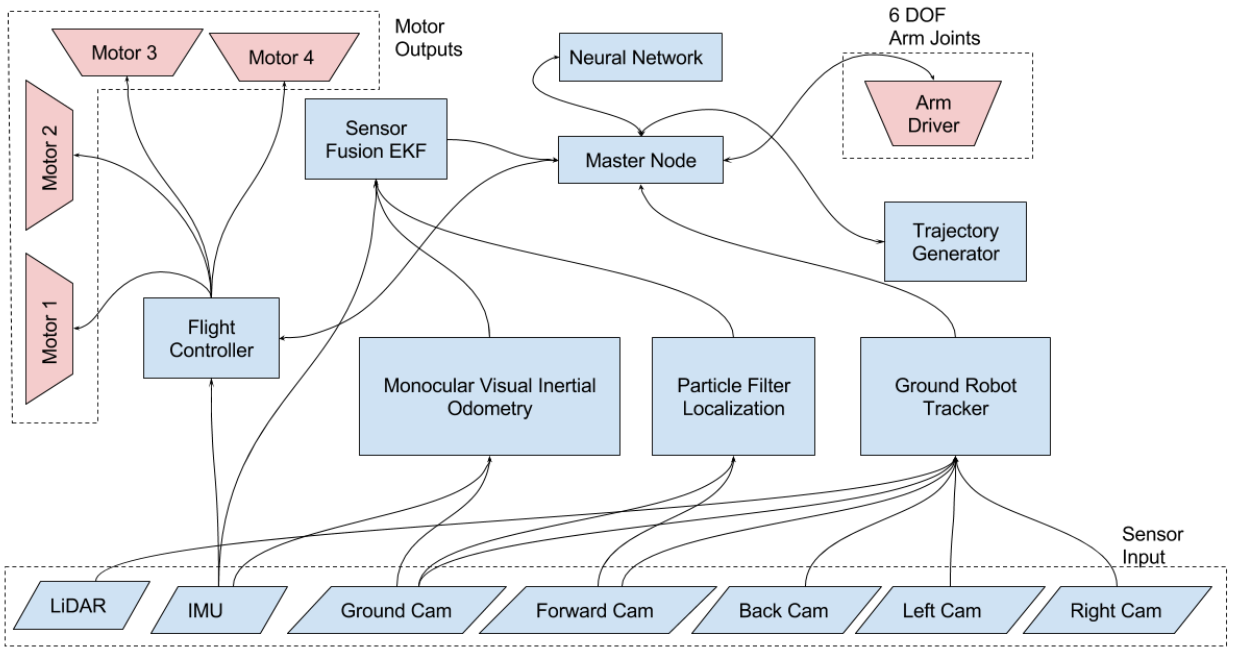

Figure 2. System Architecture

Yearly Milestones

- November 2015: Our Robotics Team was formed with the goal of participating in IARC mission 7.
- January 2016: Finished developing machine learning algorithm for the ground robot chooser
- March 2016: First funds awarded
- May 2016: Flight controller developed
- August 2016: Trajectory generation and control finished
- April 2017: Localization and tracking working
- May 2017: Initial Flight tests started

AIR VEHICLE

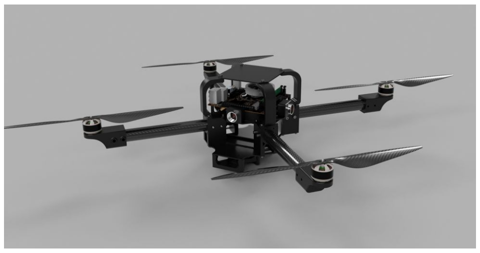

Figure 3. Airframe Rendering Without Arm

Propulsion and Lift System

We use 4 Tiger Motor MN4014 at 330 KV. We initially decided to develop a variable pitch propulsion system, but later decided not to due to time constraints. Instead, we use 4 fixed pitch 18X5.5 inch propellers from Quanum. With this configuration, each of the 4 motors can produce about 25 newtons of thrust.

For our system, it is important to know what force each motor is producing at any given time. We do this by fitting a thrust vs rpm model to experimental thrust test data. Once we have a model for thrust, we just need a way of keeping track of rpm in flight. We do this by adding a Sharp short range distance sensor in our case the GP2Y0D805Z0F.

Guidance, Navigation, and Control

All nodes were written by our team in C++ for use in ROS. The exceptions are the neural network which is written in Java and the inverse kinematics algorithm for our robot arm which was an open source ROS package.

By fusing Visual Inertial Odometry, Particle Filter Localization, and an IMU, we can robustly localize our robot in a grid-centric coordinate system. The Visual Inertial Odometry algorithm was developed in house for this competition. We named it pauvsi_vio and it is open source. This algorithm outputs nearly metric velocity and angular velocity data with a computation time of ~5 ms per frame. The Particle Filter Localization algorithm uses two cameras and reprojection error from the grid into both cameras. This algorithm takes a lot of computation time, but it returns the absolute position of our robot in the grid-centric coordinate frame and its covariance. Finally all of this information is fused together using an Extended Kalman Filter asynchronously.

The potential target ground robots are detected and classified using 5 cameras. Our algorithm works on the assumption that all potential target ground robots must stay on the floor and therefore are simply the intercept of a ray from a camera's center and the grid. We also know that each of the potential target ground robots are either red or green and within the arena. We can efficiently detect all red and green blobs in each of the camera images; then by knowing the camera's orientation, we can determine a target robot's position. Each ground robot has it's own Kalman Filter, and is updated with this information.

Obstacle robots are detected in a similar way to target ground robots, but we use the fact that each obstacle robot has a pole on it which can be detected with the lidar.

To avoid the need to touch a target robot with bottom of our airframe, we attached a 6 degree of freedom robot arm with a plate. We use this arm to touch the top of a ground robot. The joint angles for the actual touching action is computed using inverse kinematics. We use Dynamixel AX-12 servos. These servos give us feedback and allow us to set the maximum torque each can produce. This allows for a damping action which prevents significant disturbances to the actual quadrotor which is hovering at 0.5 meters during the action.

Stability Augmentation System

We employ a closed loop control system to keep our robot at desired position, velocity, acceleration, jerk, and snap. This system uses 2 PID controllers.

$$r''_{fb} = -K_P * e - K_I * \int_0^t e * \delta t - K_D * e' \tag{1.0}$$

$$e = r^i - r^i_d \tag{1.1}$$

$$e' = r'^i - r'^i_d \tag{1.2}$$

$$q_e = q^* * q_D \tag{1.3}$$

$$M_B = -q^0_e * K_P * \widehat{q_e} - K_D * (\Omega^B - \Omega^B_D) \tag{1.4}$$

Equations 1.0 - 1.4 describe the control system we employ and they are described in further detail in Cutler's paper [1]. Equation 1.0 uses position and velocity error to compute the acceleration feedback. This acceleration is then used to compute the desired orientation, q.

$$\Omega^B_{Dxy} = \overline{F} \times \overline{F'} \tag{1.5}$$

Finally, we compute the desired moment, M. To do this, we use equation 1.5 to compute the desired angular velocity, Ω^B_D. In equation 1.5, F is simply the desired acceleration plus the acceleration feedback multiplied with the mass of our vehicle.

We can compute the individual motor forces using the moment and total force which we have computed. Cutler's paper describes the linear relationship between motor forces and moments.

Navigation

Our path-planning algorithm consists of two major steps. Initially we select a target ground robot for interaction. Once chosen, we estimate its future position for waypoint generation. This algorithm then develops a goal for the quadcopter. Given this goal, we can run our minimum time polynomial trajectory generation algorithm to generate a complete trajectory.

Our waypoint generator develops a set of waypoints using three inputs:

- Current State of quadcopter
- Current State of target ground robot
- Current State of all obstacles

The state of the aerial robot consists of position and its four derivatives - up to snap. The other states simply hold a position and velocity for the target ground robot and obstacles. Each of the waypoints are generated within the environmental constraints set by the generator itself. If an obstacle poses to be an environmental constraint, the waypoints are generated above the maximum height of the collection of obstacles. These waypoints are generated at a height of 2.8m above ground level. The initial and final waypoints generated contain position, velocity, acceleration, jerk and snap, while the others just hold a position. This allows us to constrain the two end points of the trajectory to the 0th through the 4th derivative of position. This is key because the 4th derivative, snap, is related to the moments which the robot must produce along the path.

The Trajectory generator algorithm is a modified trajectory generation algorithm described by Cutler [1]. Our trajectory generator produces high order polynomials which conform to environmental, inertial, actuator, and path constraints. The algorithm fits these constraints by iteratively minimizing the time of flight for each path segment, i.e segment between a pair of consecutive waypoints. Our trajectory generation algorithm allows our robot to fly in an aggressive and agile way to each target. Finally, a 5 segment trajectory can be computed in under 50 ms with time optimization.

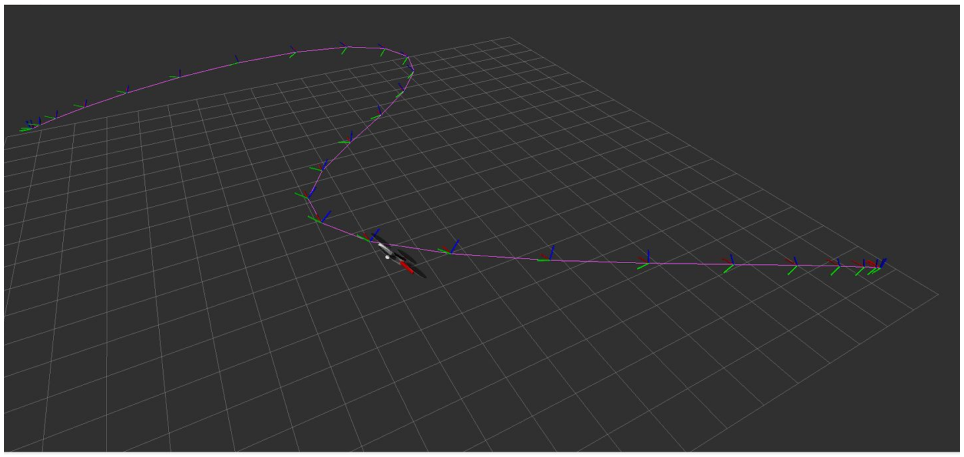

Figure 4. Simulation of Trajectory Generation and Control

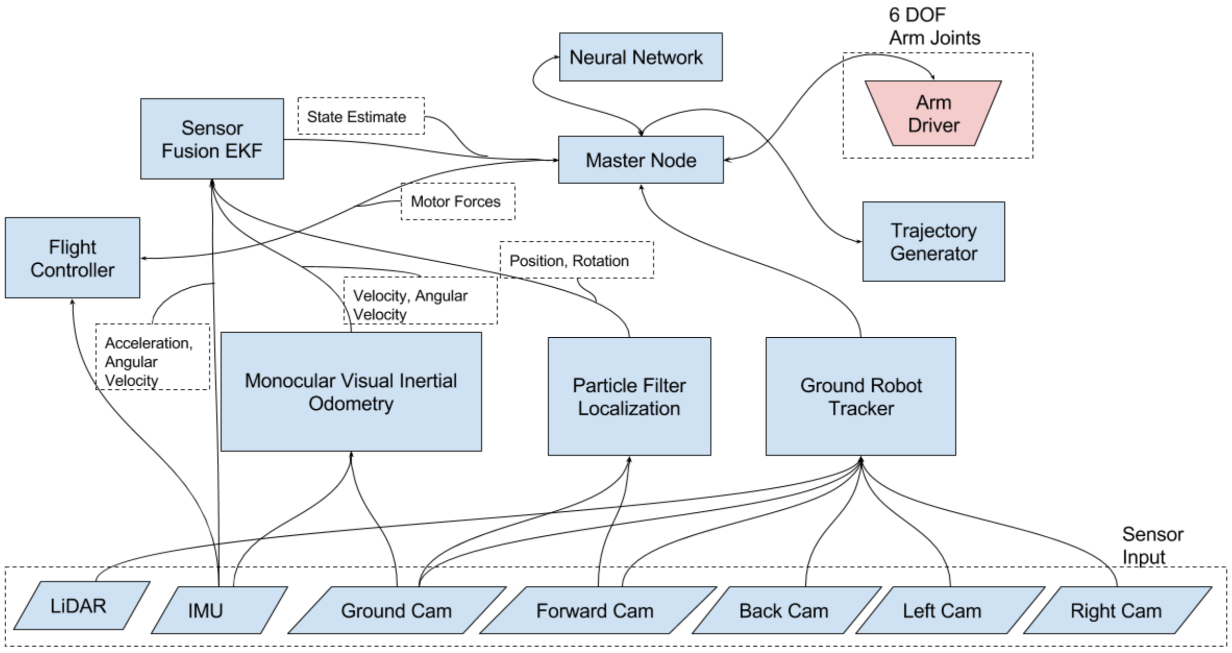

Figure 5. Figure of Control System Architecture

Flight Termination System

The termination system is designed so that there is a quick and robust way to either stop the vehicle (soft kill) or disable all motors (hard kill) in case of an emergency. It is important that this system is independent of the main computer as that is the most likely point of failure.

Our hard kill termination system uses an off the shelf RC receiver which works on 2.4 GHz. When a switch is flipped, the flight controller will immediately shut off all power to the motors. Our soft kill termination system stops the current flight and executes a landing command.

PAYLOAD

Sensor Suite

- Cameras
    - Point Grey Chameleon3 13Y3C-CS (1x)
        - 150 FPS bottom camera
    - Point Grey Chameleon3 13S2C-CS (4x)
        - 30 FPS side cameras
- IMU (Inertial Measurement Unit)
    - Invensense MPU-9250 (1x)
        - Measures linear acceleration and angular velocities
- LIDAR
    - RPLIDAR A1
        - 2D LIDAR used for threat detection
- RPM Sensors
    - SHARP GP2Y0D805Z0F (4x)
        - Measures RPM of propellers
- Rotary Encoder
    - DYNAMIXEL AX-12 (1x)
        - Measures angle of each joint on the arm

GNC Sensors

The visual inertial odometry algorithm uses the bottom camera and the IMU. The particle filter algorithm uses the bottom facing camera and the front facing camera. The potential target ground robot tracker uses all five cameras. Lastly, the close loop controller make use of the RPM sensors.

Mission Sensors

The mission sensors consist of all the sensors mentioned in the GNC sensors and in addition the rotary encoders. The GNC sensors are used to figure out the orientation of the aerial robot as well as controlling it. The purpose of the rotary encoders is to control the interaction between the aerial robot’s arm and the ground robots.

Target Identification

We used a machine learning algorithm called Neuroevolution through Augmenting Topologies [5]. This allowed us to teach a neural network to choose the best ground robot to target, and how many times to touch it.

The neural network uses the current position and velocity of both the ground robots and aerial robot as inputs, and it has an output of which ground robot to touch and how many times to touch it.

Threat Avoidance

Threats are detected using the lidar. When an unexpected interference occurs in the plane of the lidar, it is checked to see if it lies within the arena. If the interference is in the arena it is determined to be an obstacle.

Communications

We use 5.8 GHz dual channel Wifi for sending data between the ground station and robot. This gives us significant bandwidth for image transmission. We use a 2.4 GHz receiver for manual control and the hard kill termination system.

Power Management systems

We use a 6 cell 10000 mAH lithium polymer battery for our power source. We use a 250W dc ATX power supply for the onboard mini-ITX computer. We use 60A ESCs with 2A BECs at 5V. The 5V BEC powers the flight controller. Finally we use a 12V step down converter to power the AX-12 servos.

Sub-Vehicle

Our sub-vehicle is a 6 degree of freedom arm used for interact with the ground robots. This uses the rotary encoders in each of the joints of the arm.

OPERATIONS

Flight Preparations

For initial flight preparations, we ensure that the ground control station is setup and flight termination functions as expected. All preparations for flight are strictly followed sequentially in the checklist. All of the preparations are carried out by the safety pilot, along with ensuring that the quadcopter stays under all constraints throughout the round. Once the safety pilot completes the checklist and assures software and hardware functionality, the quadcopter is initiated for the round.

Checklist

Pre-initialization

- Setup ground control station (GCS)
- Ensure propellers are tightly secured and have free rotation
- Ensure all bolts are tightly screwed
- Ensure battery is charged to 25.2 V
- Ensure all connections to sensors are tightly secure
- Ensure RC transmitter turned on and functioning
- Straighten robot arm completely and ensure it is free of obstacles

Post-initialization

- Test and ensure flight termination works as intended
- Connect on board computer to GCS
- Initialize visual odometry
- Ensure vehicle state data is received by the GCS
- Ensure all sensors are up and running
- Ensure localization is fully functional

Launch vehicle for the round

Man/Machine Interface

The vehicle can be controlled by an RC transmitter and the GCS. The mission is initiated over SSH and all the vehicle state data are monitored and visualized in the GCS using Rviz and RQT, visualization and GUI tools built into ROS. The vehicle can be switched to a manual mode through a switch on the RC, allowing the team to gain complete control with the RC. The vehicle is equipped with two kill switches: a soft-kill switch from the GCS which allows for a smooth hover-landing, and a hard-kill switch from the RC transmitter which instantly kills the motors.

RISK REDUCTION

Vehicle Status

Each node in our system uses the many levels of logging built into ROS to describe the current state of the system.

Shock/Vibration Isolation

Unless vibrations cause significant noise in our state estimate, we have no need for vibration isolation.

EMI/RFI Solutions

The high bandwidth (5.8GHz) Wifi antenna is located at the bottom of the aerial robot, which is far away from all the sources of EMI/RFI.

Safety

We have a hard kill termination system, along with propeller guards to protect bystanders in case of a collision.

Modeling and Simulation

We wrote a simulation for control algorithm testing and tuning. This simulation added gaussian noise in the state estimate for realistic performance. The variance we use for our noise is higher than actual noise to ensure that the control system is strained in simulation.

In addition to sensor noise, we add thrust noise. This ensures that the system can perform even if there is a significant difference between actual and desired thrust.

Testing

At this point, testing has only been in simulation, but the results in simulation looks promising.

Our next step in testing will be actual flight tests with progressively more aggressive trajectories. We will then test the interaction between the arm and a ground robot in flight.

CONCLUSION

The architecture we have implemented displays new features that would provide us an edge over the competition. Our in-house trajectory generator allows for aggressive and agile flights unlike typical waypoint methods. Furthermore, our target selection is carried out by a trained neural network, allowing for highly optimized decision making; this performs far better than most hand written algorithms. We expect that our design choices can lead to an impressive autonomous performance.

ACKNOWLEDGEMENTS

We would like to thank our faculty advisor Dr. Inseok Hwang for giving us tips and answering some of our questions about the Extended Kalman Filter and the Particle Filter.

REFERENCES

[1] Cutler, M. J. (2012). Design and control of an autonomous variable-pitch quadrotor helicopter (Doctoral dissertation, Massachusetts Institute of Technology).

[2] Forster, C., Pizzoli, M., & Scaramuzza, D. (2014, May). SVO: Fast semi-direct monocular visual odometry. In Robotics and Automation (ICRA), 2014 IEEE International Conference on (pp. 15-22). IEEE.

[3] Bloesch, M., Omari, S., Hutter, M., & Siegwart, R. (2015, September). Robust visual inertial odometry using a direct EKF-based approach. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on (pp. 298-304). IEEE.

[4] "Official Rules for the International Aerial Robotics Competition - Mission 7," Jan 2017, http://aerialroboticscompetition.org/downloads/mission7rules_013017.pdf.

[5] Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. Evolutionary computation, 10(2), 99-127.