Development of an Outdoor Aerial Multi-Agent Robot Toolkit at the UF Autonomy Park

James W. Cross[1] and Warren E. Dixon[2]
University of Florida, Gainesville, FL, USA

This paper presents the development of a software toolkit to support ground and aerial robotics at the University of Florida's (UF) Autonomy Park, and the challenges of deploying multi-agent outdoor aerial robotics. The robots available to researchers at the UF Autonomy Park include approximately 30 robotic systems, comprised of quadruped, wheeled and multirotor platforms that need to communicate heterogeneously and run a variety of controllers to support multi-agent and adaptive control research. Outdoor localization for aerial and ground robots operating in proximity was achieved by fusing IMU data with RTK-GPS (Real-Time Kinematic Global Positioning System). Barometric-pressure-based altitude measurements varied erratically and downwards facing LIDARs were installed to maintain a constant altitude on the quadcopters. A workflow using Robot Operating System 2 (ROS2) was developed to enable researchers to implement controllers and structure programs that could be later deployed onboard the robots. An autonomous herding experiment was implemented using the author's software package and simulated results were demonstrated.

I. Introduction

Autonomous multi-agent robotic systems are in agricultural, aerospace and defense industries, where they offer increased resiliency and enhanced capabilities in dynamic environments. Multi-agent robot operations are being investigated to herd livestock between paddocks [6], where ground and aerial robotic systems are used to share livestock state information and respond appropriately. Missile intercept problems can benefit from multi-agent coordination between intercepting agents, aiming to increase the probability of intercept when targets attempt to evade [5] or eliminate interceptors with countermeasures. Multi-agent robotics combines the strengths of individual robotic systems and distributes the advantage over the whole.

There has been growing interest in performing heterogeneous experiments at the University of Florida's (UF) Autonomy Park, using aerial and ground robots outdoors in a single experiment. When developing control algorithms for a heterogeneous system, a challenge arises in the difference of each robot's available controllable inputs. For example, at the UF Autonomy Park the aerial robots have 4 controllable inputs (roll, pitch, yaw, throttle), and the ground robots have 2 controllable inputs (linear and angular velocities). Transformations are required to move a ground robot with a controller output that would be suitable for an aerial robot. The goal of this project was to develop a set of tools to allow researchers to more efficiently demonstrate novel control methods on different types of robots.

The author implemented a herding controller to demonstrate how the software toolkit can be used. Several features of the software toolkit are highlighted by the presented herding experiments. The field of indirect control, also called herding, is inspired and often compared to a phenomenon exhibited in nature when a dog is used to corral and move a herd of sheep in and between paddocks. Indirect control problems are characterized by the number of target agents, number of herding agents, interaction dynamics, and goal locations (e.g., [4]). The herding and target agents have an interaction dynamic that the herding agent can exploit to direct the target agent to a goal location. Interaction dynamics are modeled by sensor modalities that allow target agents to track herding agents. Interaction dynamics have also been used by herding agents to physically move the target agents using aerodynamic forces [4]. Applications for adaptive

[1] Undergraduate Student, Mechanical and Aerospace Engineering Department, AIAA Student Member 1342434
[2] Faculty Advisor, Mechanical and Aerospace Engineering Department

herding controllers in aerospace include redirecting animals near airports, hostile aerial swarm defense, and missile guidance [6, 3, 5].

The author's contribution was to develop a software package that accepts body-fixed velocity commands and provides state feedback in a world fixed-frame, this paradigm was developed to align with common industry standards and typical simulation environments. World fixed-frame position conversion is automatic, and all robot positions are presented to the controller in the Autonomy Park frame. The package streamlined the experimental controller implementation process by allowing the experimental controller to be written in Python and verified on any computer. The package managing the robots can execute the same code, eliminating the requirement to develop and test on a dedicated Ubuntu computer.

This paper provides relevant background about the Autonomy Park, the development of the software toolkit in Robot Operating System 2 (ROS2), and a herding experiment example. Many others contributed to bringing the Autonomy Park online and this paper would not be possible without their contributions.

II. Background

The Autonomy Park is an outdoor laboratory located adjacent to the UF campus in Gainesville, Florida. The Autonomy Park is managed by UF's Mechanical and Aerospace Engineering Department with a goal to support the development of multi-agent robotics, nonlinear controls research, robot perception research, and inter-agent communication research in real world conditions [1]. The Autonomy Park consists of a 75x27x18 meter netted enclosure that contains a small hill and grassy field. Additionally the Autonomy Park has two buildings to store robots and run experiments. An aerial view of the Autonomy Park is provided in Figure 1. The robots available to researchers at the Autonomy Park are presented in Table 1. All robots are equipped with RTK-GPS receivers for positioning data, IMUs for orientation data, and are running ROS2 Humble to network and distribute state information and execute controller outputs.

Ground Robots		**Aerial Robots**
Quadruped	**Wheeled**	
(3) Unitree Go1	(2) Clearpath Jackal	(1) Freefly Alta-X
(1) Unitree B1	(2) Clearpath Husky	(3) Freefly Astro
		(20) Custom[3] Multirotor

Table 1: Robots available at the Autonomy Park

Fig. 1: Aerial view of the Autonomy Park netted enclosure, control buildings can be seen at the end of the enclosure [1].

[3] "Custom" refers to lab-built quadcopters assembled using a variety of off-the-shelf parts.

III. Technical Challenges and Solutions

A. Localization Challenges

The Autonomy Park facilitates multi-agent experiments with the option for over 20 vehicles being independently controlled simultaneously. Accurate localization is necessary to evaluate controller performance and safely control multiple robots in a confined area. Inaccurate sensors may cause reported robot positions to shift erratically by greater than several times a robot's average length, even if the robot is not moving. Collision avoidance and potential field boundary programs are used to minimize the risk of a collision, which both rely on accurate position estimates to correct hazardous trajectories. Accurate position estimates are critical for robotics and especially for outdoor robotics where nonlinearities caused by the environment can influence an experiment.

B. Obtaining Latitude and Longitude

An RTK-GPS system has been successfully used for accurate localization within the Autonomy Park. Uneven heating of the troposphere and charged particles from the sun in the ionosphere alter the speed of signals from GPS satellites. The result is that for a given location, the reported value from a GPS receiver will vary an unacceptable amount for proximity-based operations. RTK-GPS mitigates this error as the technology consists of a base station located at a fixed location broadcasting measurement corrections to compatible receivers mounted on the robots. At the Autonomy Park, an *Emlid Reach RS3* RTK base station is attached to a tower adjacent to the netted enclosure. An accurate position reading was obtained by recording approximately 24 hours of GPS data with the receiver attached to the tower and processing the data using a tool such as CSRS-PPP [2].

C. Maintaining Altitude

One consequence of the Autonomy Park's north Florida location is that summer storms can form and move inland over the course of a day. The aerial vehicles at the Autonomy Park rely primarily on barometric pressure to determine altitude. A consequence of rapidly developing summer storms is that atmospheric pressure at ground level can vary by up to 3 kPa. During these events, multirotor aircraft have been observed accelerating upwards into the netted enclosure while attempting to hold a stable position. The altitude discrepancy is believed to be the result of a sufficient difference in the AMSL pressure of the region with the actual local barometric pressure.

Solutions to this problem are ongoing and include a LIDAR coupled with a height map and meteorological equipment to offer corrections. Single beam LIDARs have been attached to the multirotor aircraft and pointed directly downwards to measure the distance to the ground. Onboard flight computers are set to maintain a fixed height above the ground. The main limitation with this approach is that aerial robots cannot fly over ground vehicles or other aerial robots as it would result in erroneous distance readings from the LIDAR beam contacting the robot instead of the ground. Additionally, altitude estimates lose accuracy when the vehicle is pitching forward and the LIDAR beam is no longer pointed directly downwards.

Another approach being investigated includes mounting an ultrasonic anemometer and barometer unit on a nearby tower. These sensors can record wind speed and direction along with atmospheric pressure to share with the robots along with allowing operators to determine if conditions are safe to launch an experiment. This system is still in development and will be used in conjunction with LIDAR readings.

IV. Software Framework Design

The presented software toolkit serves to aggregate and fuse sensor data to generate state information for the intended robot platform, in a common reference frame amongst all robots. The state information is used by the experimental controller to process the control outputs. The product is a software suite that facilitates the input of state data to controllers and handles the output of velocity messages using the same data types and overall code structure in simulation and on various robots. More specifically, ground robots with fewer degrees of control than aerial robots should be able to receive the same commands from a controller.

ROS2 is a middleware that runs the Autonomy Park robots allowing a single node to interface with multiple robots. ROS2 uses a network of *topics* to distributively share information, eliminating the need for a central computer to send individual commands to every robot. ROS2 packages are most commonly developed using the Ubuntu operating system, which may not be a researcher's native operating system. The author's software toolkit is comprised of the following parts:

1) Python Simulation
2) ROS2 Simulation
3) ROS2 Experiment Package

A. Core Components

The python simulation is designed to allow researchers to convert controllers written in other languages to python code that uses the same variable names and program structure as common ROS2 nodes. A matplotlib window is included to visualize trajectories calculated by integrating the controller output using a common Euler integration method. The python simulation is not intended to replace tools used for controller evaluation, but to verify correct implementation in a format that can be easily copied into ROS2.

The ROS2 simulation runs the same code from the python simulation and verifies implementation in a ROS2 environment. An RVIZ visualization, in addition to a 2d plot, is generated when the controller is running. For the herding example, the visualization is composed of *herding agent* and *target agent* nodes along with an integrator and visualization node. The nodes include common functions that influence dynamics, behavior and evasion dynamics to customize the characteristics of each robot.

The ROS2 experiment code modifies the ROS2 simulation package by removing integrator nodes and routing output velocity commands directly to the robots. Robot position estimates are drawn from the fused state data and passed into the controller. Additionally, key values such as target agent tracking error, position, and velocity are saved to a CSV file for post-experiment analysis.

B. Functions

A number of functions are included to aid in controller implementation. The software toolkit was designed to initially support a herding experiment which consists of a herding agent directing a target agent towards a goal location. The herding agent and target agent nodes contain relevant functions and are listed in Table 2. Both nodes also subscribe to other robot positions and can publish commands to the onboard motor driver.

Herding Agent Node	Target Agent Node
convert_velocity()	convert_velocity()
distance()	distance()
controller()	interaction_function()
	levy_walk()

Table 2: Herding agent and target agent node functions included in the toolkit.

The wheeled ground vehicles at the Autonomy Park are either mechanically differential-drive robots or lack individually addressable motor control. Effectively, all the wheeled ground vehicles can accept only a forward or backward velocity and an angular velocity about the vertical axis. The *convert_velocity* function converts nonholonomic velocity commands from a controller into linear and angular velocities for the ground vehicles. A Euclidian norm and atan2 of the controller output velocity are used.

The primary benefit to this approach is that researchers can treat each robot as interchangeable agents, sending the same velocity messages to quadcopters and differential drive robots. This approach effectively treats ground robots as holonomic agents in a heterogenous experiment. Limitations of this approach include how the software toolkit currently only works with 2d simulations, i.e. the quadcopters do not change altitude and all robot positions are projected onto a plane parallel to the ground. Additionally, the program is limited to controllers that output velocity commands. Because of this limitation, 2^{nd} order control is not directly available. Future goals for the project include allowing experimental controllers to modulate quadcopter altitude and to develop software drivers to allow acceleration control of the robots directly or numerically from sensed and filtered velocity information.

Distance is calculated using the Euclidian norm of the positions of each robot. The *distance* function is calculated within the Autonomy Park frame. The Autonomy Park frame is defined as a Cartesian space with the origin centered in the middle of the park on the ground. To accomplish a simplified coordinate space in the real world, the ROS2 transform (TF) module was used to transform reference frames and convert sensor data onto the centroid of the robots and ultimately express the position and orientation of each robot to the geometric center of the Autonomy Park. The use of a common reference frame simplifies experiment design, as goal locations and expected trajectories are expressed in XYZ values. The robots natively receive position information from RTK-GPS receivers, but then the data is fused with IMU and odometry data using the Robot Localization ROS2 package. The TF module is used within the robot localization package to handle the transformations to the Autonomy Park reference frame.

The *controller* function contains the controller being tested and is provided with every robot's position and outputs velocity commands either directly to a robot or to *convert_velocity* first. By running a separate node for every agent, the controller is able to run onboard each robot. Onboard processing is critical to several types of planned experiments which incorporate jamming and uncertain state information, where the robot must still be able to execute its mission without a connection to a central computer.

The *interaction_function* is critical to planned herding experiments where target agents are herded towards a goal location using some interaction dynamic. The function is currently designed to cause the target agent to move in the opposite direction of the herding agent when the *distance* function is below a given threshold. Future improvements include adding uncertainty into the target agent response.

Additionally, it may be desirable for target agents in a herding experiment to move in a general area when not being actively herded. Loitering is controlled with the *levy_walk* function, which uses a pareto distribution to output small linear and angular velocities, such behavior is analogous to animals in the real world that may wander when left unattended.

V. Implementation Example

Figures 2 and 3 depict a single agent tracking a single target agent with first order dynamics, shown in (1), using a simple herding controller. The error of the target agent with respect to the goal location is denoted with e_x, and the error of the herding agent with respect to the desired trajectory is denoted by e_y. User selected gains include k_1 and k_2, while g represents the interaction dynamic which is currently a constant.

$$u \triangleq k_2 e_y - \left(\frac{k_1^2}{g} + g\right) e_x \tag{1}$$

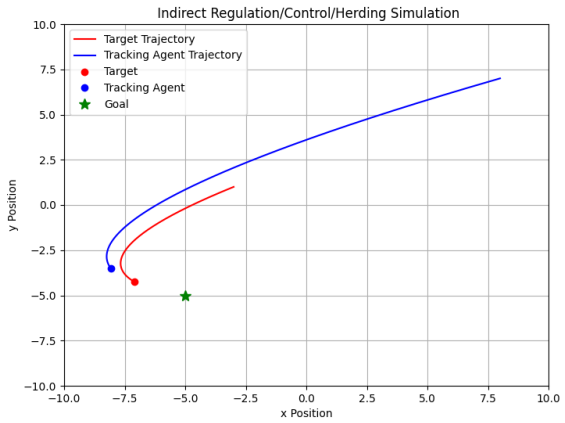

Fig. 2: Trajectories of a herding simulation

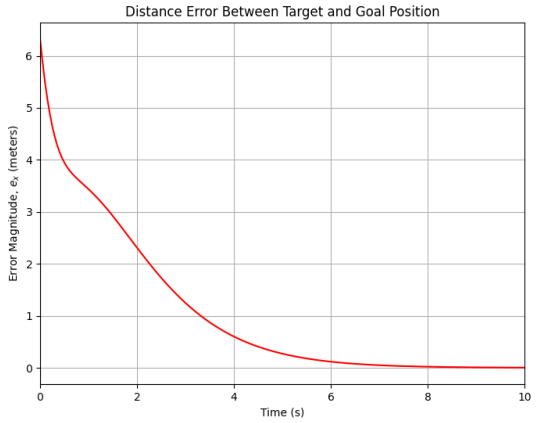

Fig. 3: Target agent goal distance error, e_x (meters)

Figure 4 depicts agent trajectories as position and velocity data is shared between nodes using ROS2. Figure 5 uses a ROS2 tool called RVIZ to plot the location of poles supporting the nets and the origin of the Autonomy Park coordinate frame at the center of the grid. Additionally, vectors that indicate which direction the robots are facing are plotted in real time. Coupled with the ROS2 simulation code, a rough estimate of the trajectories can be viewed in Euclidian space to determine if the goal or starting locations are placed in unreachable areas or close to the net.

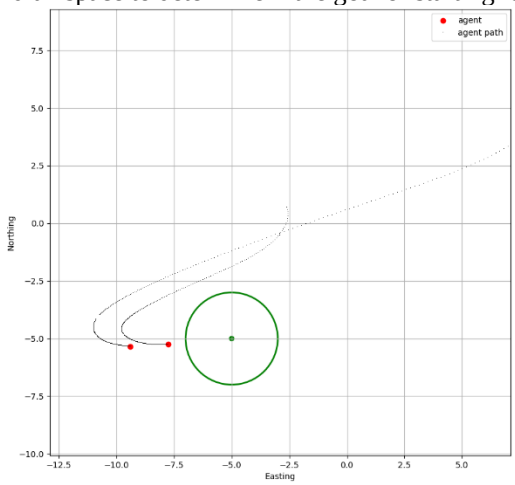

Fig. 4: Simple herding problem 2d visualization in real time using data shared between ROS2 nodes

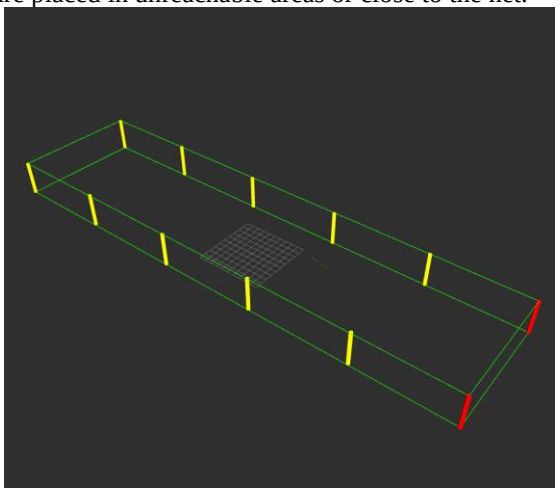

Fig. 5: RVIZ visualization of the Autonomy Park enclosure

Figure 6 uses a ROS2 development tool called RQT-Graph to plot the topics and nodes active in the simple herding experiment. The ellipses in the top left corner are from the ROS2 TF library and are the transforms used to convert from global GPS positions to the local Autonomy Park frame. The boxes in the center correspond to the herding agent and target agent robots and indicate how position data is shared between robots and internally.

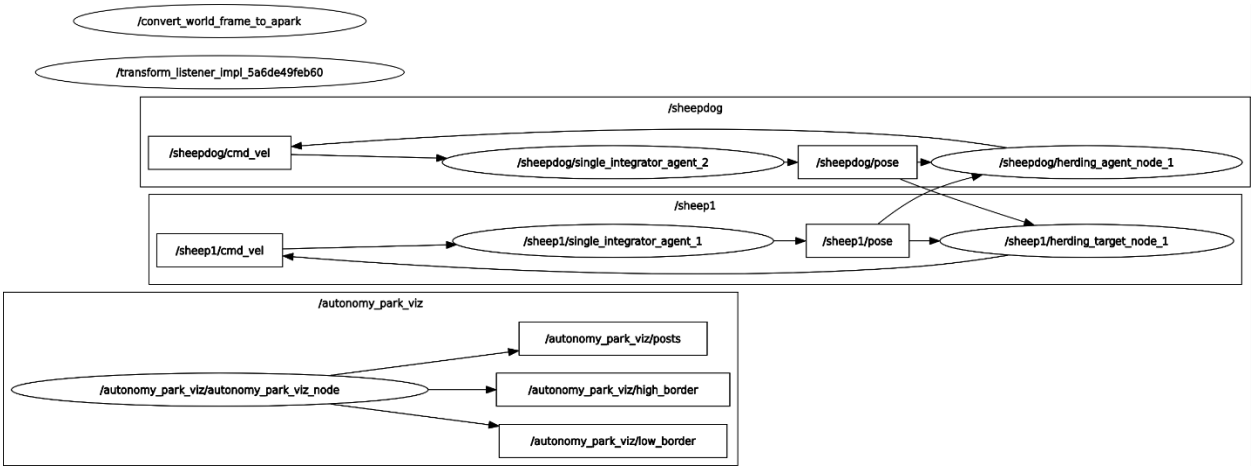

Fig. 6: RQT-Graph plot of active ROS2 nodes, topics, frames, and subscribers

VI. Conclusion

This paper presents the challenges in implementing multi-agent outdoor robotics at UF's Autonomy Park along with the author's software toolkit designed to facilitate experiments. Precise outdoor localization of multiple robots in proximity-dependent operations is necessary for safe and accurate experiments. The development of solutions to inconsistent altitude control are ongoing. The author's software toolkit allowed ground robots and air robots to receive the same types of output from a controller and function as interchangeable robots with a set of useful functions for experiments. Finally, the author's software toolkit was tested using a controller to demonstrate sample results for a multi-agent herding experiment.

Acknowledgments

The author would like to acknowledge Jhyv Philor, for developing a simple herding controller to conduct an experiment. The author would also like to thank Patrick Amy, Dr. Sage Edwards, William Warke, Brandon Fallin, and Jordan Insinger for their help in solving challenges at the Autonomy Park.

References

[1] Autonomy Park – Department of Mechanical & Aerospace Engineering. https://mae.ufl.edu/research/facilities/autonomy-park/. Accessed Feb. 20, 2025.

[2] Canadian Geodetic Survey. CSRS-PPP: Precise Point Positioning. https://webapp.csrs-scrs.nrcan-rncan.gc.ca/geod/tools-outils/ppp.php. Accessed Feb. 25, 2025.

[3] Chipade, Vishnu S, and Panagou, D. "Aerial Swarm Defense Using Interception and Herding Strategies." *IEEE Transactions on Robotics*, Vol. 39, 2023, pp. 3821–3837. https://doi.org/10.1109/TRO.2023.3292514.

[4] Licitra, R. A., Bell, Z. I., and Dixon, W. E. "Single-Agent Indirect Herding of Multiple Targets with Uncertain Dynamics." IEEE Transactions on Robotics, Vol. 35, 2019, pp. 847–860. https://doi.org/10.1109/TRO.2019.2911799.

[5] Licitra, R. A., Neale, A. J., Doucette, E. A., and Curtis, J. W. Adversarial Aircraft Diversion and Interception Using Missile Herding Techniques. No. 10982, 2019, pp. 335–343. https://doi.org/10.1117/12.2519148.

[6] Long, N. K., Sammut, K., Sgarioto, D., Garratt, M., and Abbass, H. A. "A Comprehensive Review of Shepherding as a Bio-Inspired Swarm-Robotics Guidance Approach." *IEEE Transactions on Emerging Topics in Computational Intelligence*, Vol. 4, 2020, pp. 523–537. https://doi.org/10.1109/TETCI.2020.2992778.