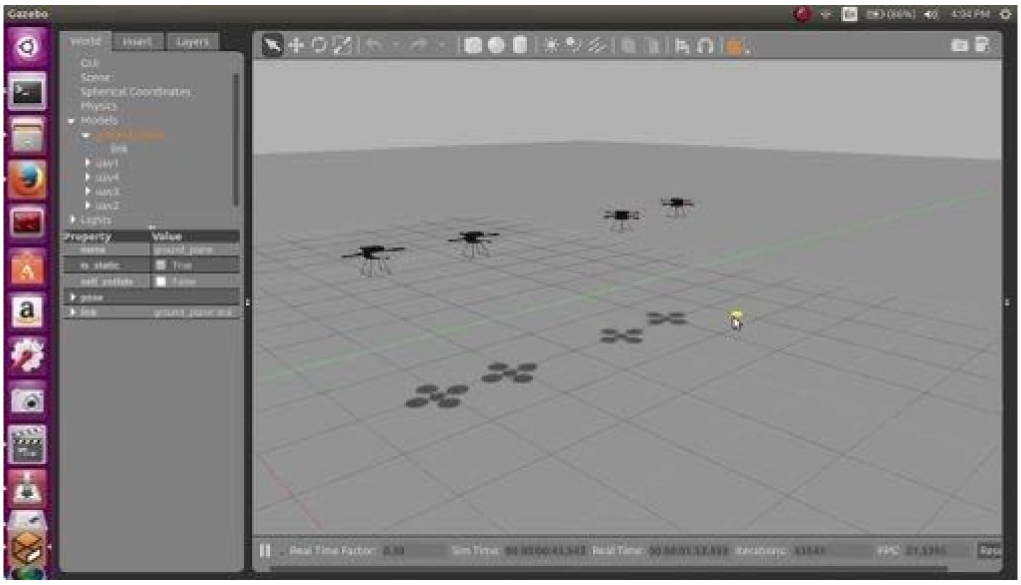
Swarm Simulation and Search using Gazebo/ROS

Two credit mini-project



Team:

- Aravind S (PES1201801734) - ECE 5D
- Pranay Mundra (PES1201801166) - ECE 5A

Project Guide:

Dr. T.S. Chandar
Professor, Dept. of Electronics and Communication

Session - August-December 2020

Abstract:

With a growing need for search operations requiring covert tracking of an object/individual while maintaining a lack of human presence in hard-to-reach regions, drones equipped with cameras and laser scanners are becoming increasingly popular in this regard. We present a simulation and a proof of concept using the `hector_quadrotor` package to demonstrate swarm capabilities such as efficient search, communication between master and slave(s), and a leader follower network. All these together enable us to design a system where a drone locates an object, following which it becomes the master and directs other drones in the network to gather around the object while remaining in formation.

Contents:

1. Motivation:

Artificial swarm intelligence may be achieved by a desired collective behavior that emerges from the interaction between robots and interactions between the robot and the environment. This fact can be used for many practical purposes where humanoid autonomy is required such as search and rescue missions, pesticide control on large farms, etc. A necessary step in heading towards this autonomy is the task of searching for and locating an object/individual as well as continuously tracking it once it is found.

2. Literature survey:

The research papers glanced through so far:

- J. Leonard, A. Savvaris, A. Tsourdos, Towards a Fully Autonomous Swarm of Unmanned Aerial Vehicles, 2012, Proceedings of 2012 UKACC International Conference on Control

  This paper proposed the implementation of a drone swarm in cities whose layout is known. Algorithms for path planning and optimal destination assignments were discussed.

- V. Sulak, I. Kotuliak, P. Cicak, Search using a swarm of unmanned aerial vehicles, 2017, 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)

  The paper introduces the different path planning and control approaches for a drone swarm as well as search algorithms to minimize costs in the graph. Different network architectures that could be used for a swarm were also discussed.

- M. Gotzy, D. Hetenyi, L. Blazovics, Aerial Surveillance System with Cognitive Swarm Drones, 2016, 28th International Conference on Cybernetics and Informatics

  This paper discusses higher level logic involved in area surveillance as dispersion behaviour, target interception and circulation.

- H. R. M. Sardinha, M. Dragone, P. A. Vargas, Closing the Gap in Swarm Robotics Simulations: An Extended Ardupilot/Gazebo plugin, 2018

  This document discusses and implements a new plugin in the ROS/Gazebo framework for swarm based simulations and visualizes the geometric centre of the swarm.

3. Problem statement:

To simulate a scenario where multiple drones work together to locate a target in an arena defined by a floor plan and to surround it in formation. A swarm of four drones shall be deployed over a given area. Each drone in the swarm shall have its area to monitor. Once a drone detects the object, all the other drones shall gather around the object in a particular formation (circle, square, etc). If the object moves, the drone swarm shall follow the object while maintaining the fixed formation.

4. Use cases:

- Search and Rescue operations
- Remote tracking and surveillance
- Covert military operations
- Pesticide control on farms

5. Workflow:

I. Setting up Gazebo and ROS and going through the tutorials. Installing the Quadrotor packages.
II. Familiarity with the `hector_quadrotor` package and implementing a PID controller to move a drone from point A to B.
III. Building an algorithm to make a swarm of 2 drones, such that one drone always follows the other.
IV. Extending and scaling the above algorithm to many drones (Leader follower network).
V. Building the world in which all of this shall take place(floor plan, etc).
VI. Making a movable colored object for the drone swarm to follow.
VII. Making an algorithm to detect the colored object and follow it maintaining a constant distance from it.
VIII. Building an algorithm to have the drones always follow a fixed path whose cost is minimized compared to a random path.
IX. Implementing a procedure such that, on detection of the object by any one drone, it alerts the others and all of them gather around that place in a given pattern.
X. When asked to, the drones should break off and go back to their original paths.
XI. Implementing obstacle avoidance onto all drones, so that they don't collide with one another.

6. Deliverables:

- A complete simulation of a swarm of drones in gazebo
- The drones shall be able to locate a designated target through visual or other sensory means
- Once a target is located, all drones shall arrive there and stay in formation while following the target
- Each drone shall exhibit obstacle avoidance to prevent collision with obstacles

7. Software required:

Our project was implemented solely on systems running Ubuntu 18.04 LTS

Languages:

- Python - `rospy, openCV, mlrose, numpy`
- C++ - `roscpp`

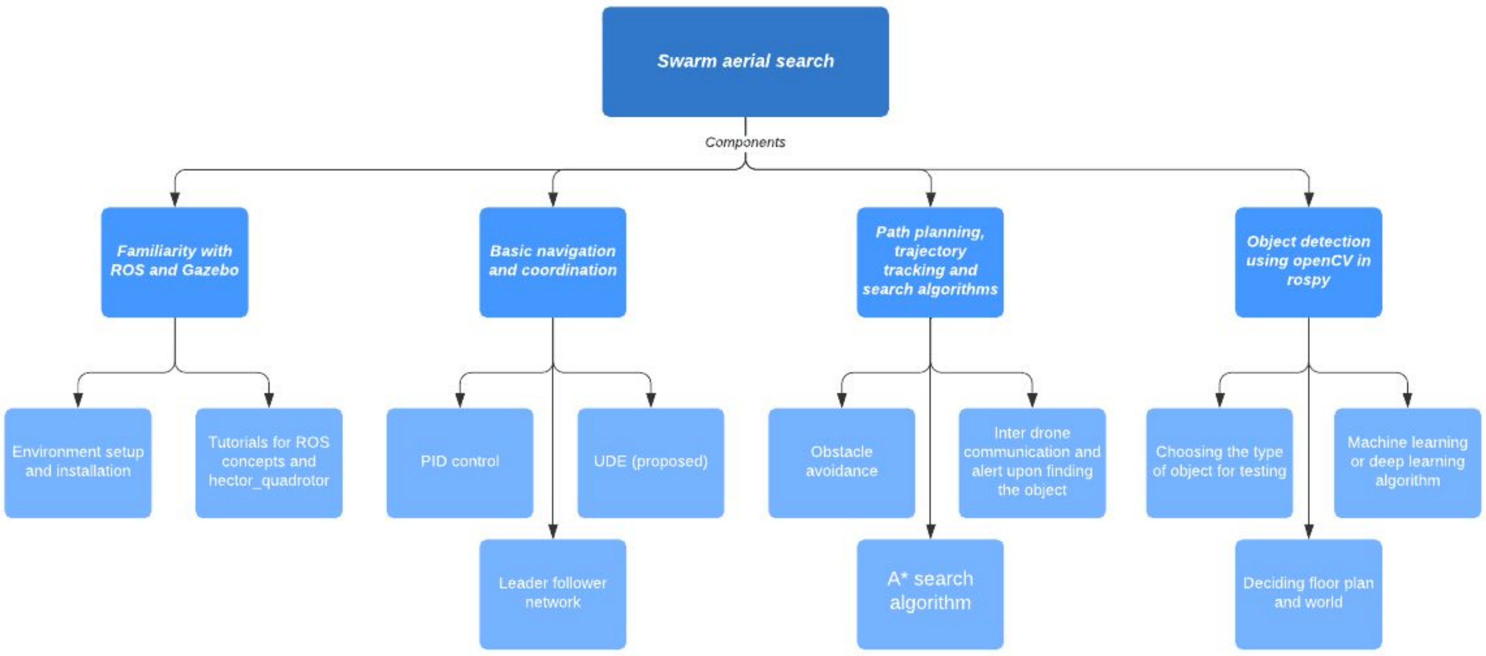
Frameworks and simulators:

- ROS Melodic
- Gazebo 9.0

8. Hardware used:

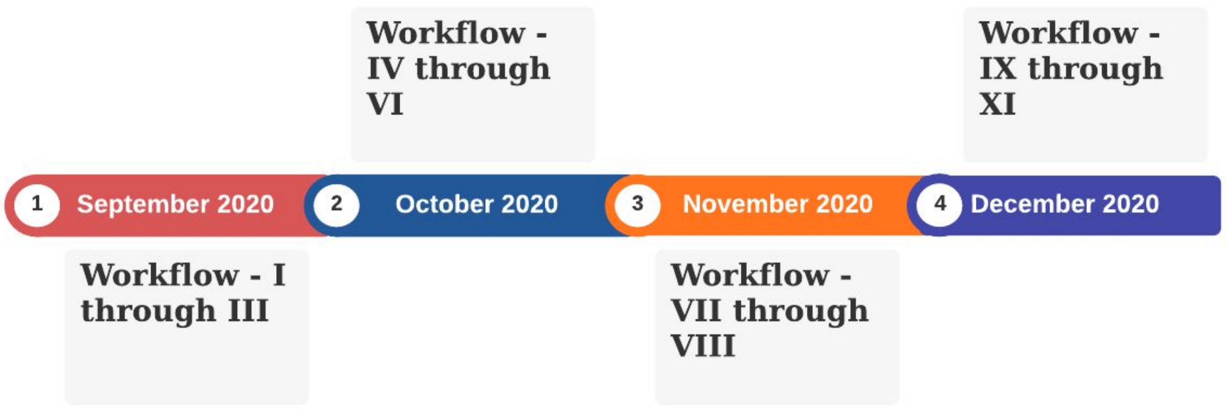
NIL

9. Concept map:



Swarm aerial search
Components
Familiarity with ROS and Gazebo
Basic navigation and coordination
Path planning, trajectory tracking and search algorithms
Object detection using openCV in rospy
Environment setup and installation
Tutorials for ROS concepts and hector_quadrotor
PID control
UDE (proposed)
Leader follower network
Obstacle avoidance
Inter drone communication and alert upon finding the object
A* search algorithm
Choosing the type of object for testing
Machine learning or deep learning algorithm
Deciding floor plan and world

10. Timeline:



11. PID Controller and waypoints:

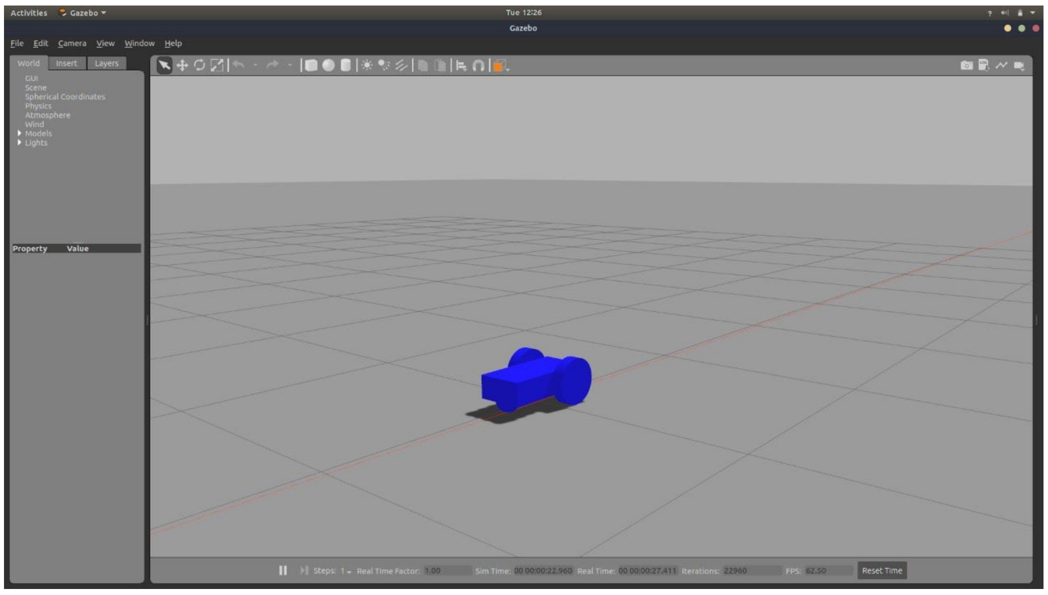
A PID controller was implemented using the ROS architecture which allows the drone to move between two static points (x_1, y_1, z_1) and (x_2, y_2, z_2). The controller node *subscribes* to the topic `/ground_truth_to_tf/pose` in the drone's namespace to retrieve the current position in the global frame. This value is compared with the destination and the errors calculated are used in the PID loop to find the velocity to be *published* to `/cmd_vel`. The demo can be found here. Following this, navigating through a predefined set of waypoints was desired. Accomplishing this task was relatively simple once the PID controller was tuned for better performance. A queue containing the points is maintained where points are dequeued upon reaching it within a constant fudge factor.

12. Leader follower network:

A leader is defined to be the drone which sets the next waypoint and instructs other drones in the network to follow or surround it in a predefined fashion such as maintaining a polygon formation with the leader at its center. This is particularly important upon detecting an object in our case. Each drone *subscribes* to a user defined topic `/leader_info` which contains information about the current leader or the topic isn't being published if a leader is not set, as is the case at the beginning of the search when the object is yet to be found. A user-defined message `leader.msg` on this topic is used to signal each drone in the swarm who the current leader is. The relative spacing between the drones with respect to the leader is then defined and a PID loop is sufficient to maintain the separation. A demonstration for the same can be found here.

13. Designing the bot:

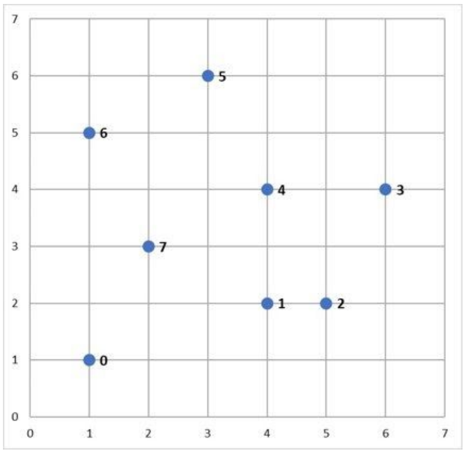
The bot here is a simple car, with two back wheels and one caster wheel. The description of the car was written in SDFormat (Simulation Description Format). SDFormat, sometimes abbreviated as SDF, is an XML format that describes objects and environments for robot simulators, visualization, and control. After the robot was modelled in SDF, its motion had to be described. This was done by writing a plugin in C++, which defines the velocities and forces to be applied on the bot so that it can move within the simulation environment. The plugin was connected to the ROS interface so that it can accept velocity values from the user.

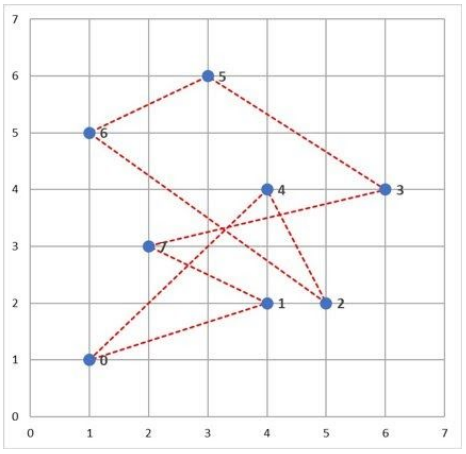


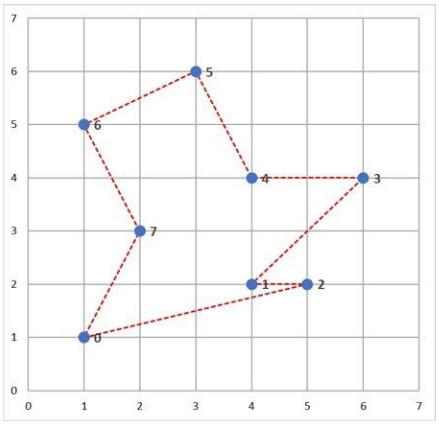
14. Solution to the Travelling Salesman Problem:

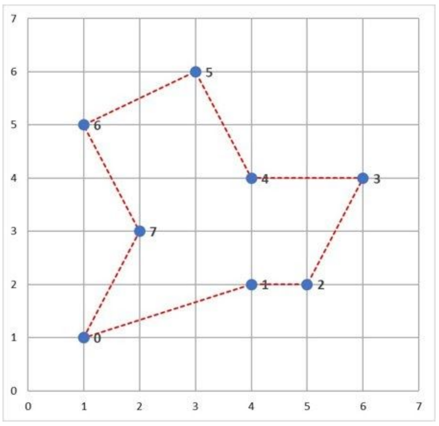
The travelling salesperson problem (TSP) is a classic optimization problem where the goal is to determine the shortest tour of a collection of n “cities” (i.e. nodes), starting and ending in the same city and visiting all of the other cities exactly once.

The ideal solution to the TSP involves generating all possible combinations of paths in the graph and finding the one which incurs the least cost(total distance). This grows exponentially as *O(n!)* which is not practically feasible. Instead, We use a randomized solver from a fork of `mlrose` which greatly reduces the time taken to find a reasonably short path in the search grid. The set of points are randomly generated in each quadrant of the grid and assigned to a specific drone followed by defining the fitness object that takes in the set of points and the euclidean distance between each pair. This object is handed off to the genetic solver that returns a solution reasonably close to the global minimum. The bottleneck hence remains to be the function that computes the distance between each pair of points. This gives us an overall complexity of $O(n^2)$.


Points in the search grid

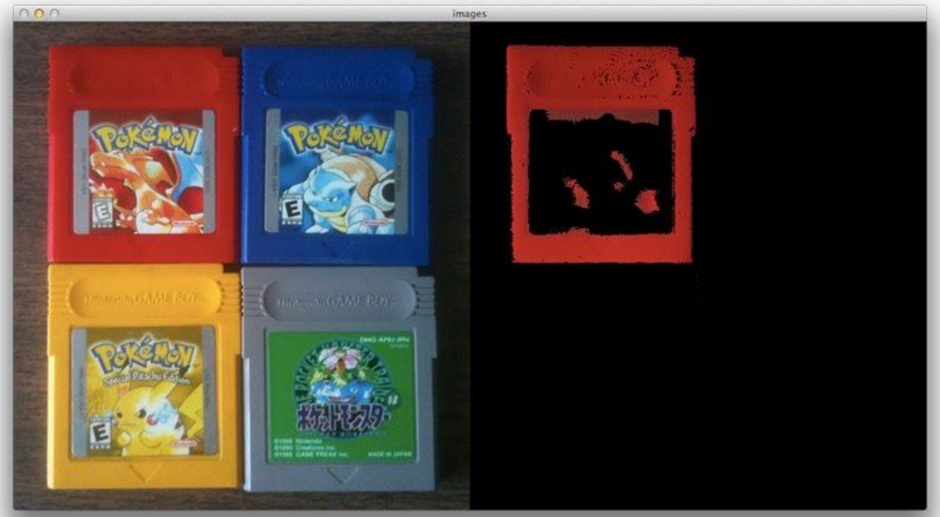

A random tour of the graph


Path from randomized solver


Ideal path from brute force search

15. Object detection:

The bot used for the purpose of detection is blue in colour so a mask using the `cv2` library is sufficient for finding the region contained by the object within each frame of the camera stream. `cv_bridge` is for converting the image in `sensor_msgs/Image` to a format compatible with the methods in OpenCV. A mask is first created for blue coloured objects using `cv2.inRange` followed by finding the contours around the mask using `cv2.findContours`. The contour with the largest area is chosen and a minimum enclosing circle is drawn around it. The parameters chosen for the PID controller in order to track the object include the center and radius of this enclosing circle. The `radius_error` is used to control the velocity of the drone in the x-direction whereas the `center_error` is used for controlling the velocity in the z-direction as well as the drone's yaw.



An example of a red colour mask in an image. Smaller regions outside of our interest can be removed using a filter such as a Gaussian filter.

16. Conclusion:

The potential for drone swarms to aid in search operations while simultaneously increasing its efficiency is certainly a worthwhile endeavour. The use cases extend beyond just searching towards fields like agriculture, defense, mapping and planning. The hurdles involved in such an undertaking include tuning the controllers for the drone, planning short routes in the search grid as well as the image processing required for detecting objects/humans. We first studied the ROS architecture and became familiar with Gazebo following which we implemented path planning and object detection using `hector_quadrotor` as the base package for our drone model. The leader follower network was then established upon locating the bot. This instructed the others in the swarm to assemble in formation around the leader, hence terminating the search.

17. Future Work:

- A number of improvements can be made to our model starting with an improved controller for movement and tracking. Using a more robust controller in place of PID can result in better response times and stability.
- A choice exists between the center of the minimum enclosing circle and the centroid of the contour drawn on the object for the purpose of tracking.
- Creating a complex environment which estimates real world scenarios more accurately.

- Implementing object and collision avoidance between each drone and its environment is an important next step.
- Refactoring the higher level logic into separate nodes serves to create a usable navigation stack. This also indicates that we would be benefited if we were to use the standard navigation stack in ROS developed by Willow Garage.

18. References:

1. J. Leonard, A. Savvaris, A. Tsourdos, *Towards a Fully Autonomous Swarm of Unmanned Aerial Vehicles*, 2012, Proceedings of 2012 UKACC International Conference on Control
2. V. Sulak, I. Kotuliak, P. Cicak, *Search using a swarm of unmanned aerial vehicles*, 2017, 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)
3. M. Gotzy, D. Hetenyi, L. Blazovics, *Aerial Surveillance System with Cognitive Swarm Drones*, 2016, 28th International Conference on Cybernetics and Informatics
4. H. R. M. Sardinha, M. Dragone, P. A. Vargas, *Closing the Gap in Swarm Robotics Simulations: An Extended Ardupilot/Gazebo plugin*, 2018
5. Q. Cui, P. Liu, J. Wang, J. Yu, *Brief Analysis of Drone Swarm Communication*, 2017, IEEE
6. C. Bian, Z. Yang, T. Zhang, H. Xiong, *Pedestrian Tracking from an Unmanned Aerial Vehicle*, 2016, IEEE
7. S. Waharte, S. J. Julier, *Coordinated Search with a Swarm of UAVs*, 2009

19. Appendices:

The resources and tutorials referred to:

- ROS Tutorials by Open Source Robotics Foundation
- Lectures by ETH Zurich
- `hector_quadrotor` Tutorials
- Pixhawk 4 Gazebo Plugin
- Randomized solver for the TSP using `mlrose`
- OpenCV tutorials for masking and contours

The code developed for the simulation and `.launch` files can be found in our **GitHub repository**.