Entropy Based Distributed Behavior Modeling for Multi-Agent UAVs Software Documentation

Created: 6/25/2021

Edited: 9/2/2021

Documentation by:

Jason Carnahan

Software Written by:

Jason Carnahan

Luke Fina

Table of Contents

System Overview 3

Relevant Terms and Acronyms 3

- System 3
- Names with <> 3
- Static 3
- Instantiate 3
- *.py 3
- ABC – Abstract Class 3

Preliminary Class Diagram 4

- CentralControl 5
- UAV 5
- <Strategy> 5
- AirSimStrategy 5
- TelloStrategy 5
- Calculations 5
- Management 5
- GUI 5

Class Outline 6

- CentralControl 6
- UAV 8
- Management 9
- Calculations 9

<Strategy> 10

AirSimStrategy 10

Changes from Luke's Code 12

- Figure 1 12
- Figure 2 13

System Overview

This software is designed to simulate the actions of a swarm of autonomous UAVs. The goal of the swarm is to move to a series of waypoints while maintaining an entropy value below an arbitrary threshold. Each UAV will calculate its own entropy then determine one of a few actions. These conditions are checked continuously as the simulation runs.

Relevant Terms and Acronyms

System

Any reference to the term “system” represents the full environment including the application, the objects developed within it, hardware, and any relevant simulations.

Names with <>

Angle brackets represent interfaces in the software design. Interfaces allow multiple implementations for expansion.

Static

Any variables described as being “static” do not change for the duration of the simulation.

Instantiate

Program creates an in-memory object of the described class.

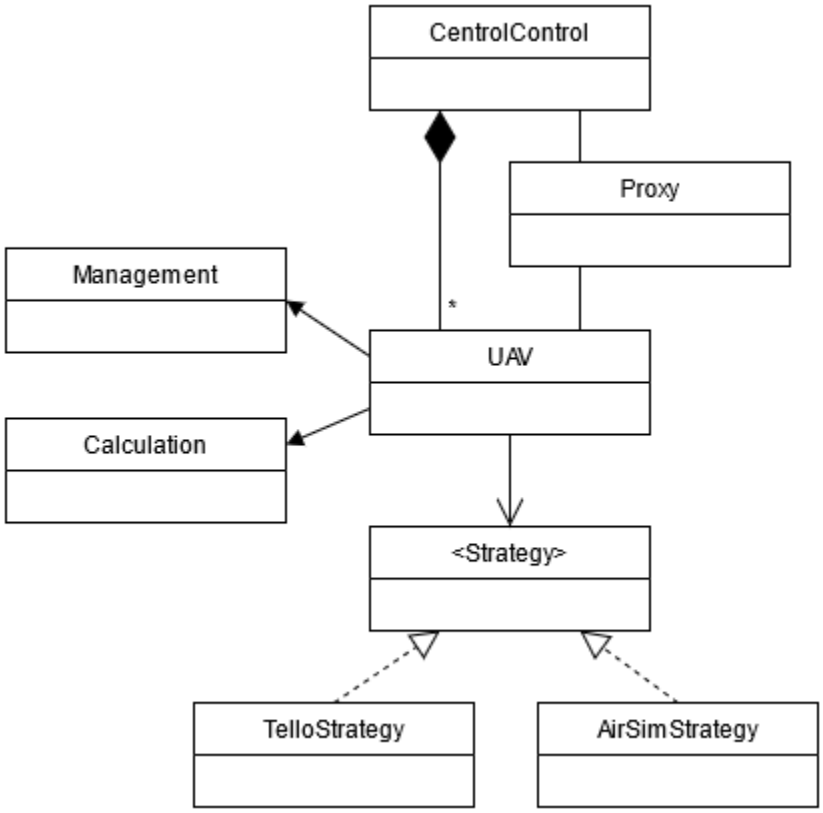
*.py

These are the Python files for the program.

ABC – Abstract Class

Abstract classes can be referred to as an ABC. These classes lay the foundation of other classes but cannot be instantiated themselves.

Preliminary Class Diagram



CentralControl

The CentralControl class controls any meta data manipulation such as image processing and data analytics. This class will report and keep track of the current UAV formation, but calculations are handled by individual UAVs. Here scenarios can be pre-programmed for the UAVs to perform.

CentralControl will (eventually) communicate to the UAVs via the Proxy. Currently, this proxy is not implemented thus the CentralControl communicates directly to the UAVs. The Proxy will be developed to facilitate different communication methods between the CentralControl and the UAVs such as radio or in-memory communication.

UAV

UAV class represents the individual UAV in the system. The UAVs will use its Strategy to communicate with any other UAV in the system and the CentralControl. Each UAV will perform its own calculations with its onboard computer as well as handling self-govern decision making. It can accept commands from the CentralControl and use its own Strategy to move in the environment.

<Strategy>

This abstract class will be the base interface for all future implementations of UAV communication and movement.

AirSimStrategy

This class implements the Strategy abstract class for use specifically in the AirSim Unreal Simulation.

TelloStrategy

This class implements the Strategy abstract class for use specifically in on Tello Drones.

Calculations

Calculations will perform any processing for the UAV such as distance and entropy calculations. These calculations will aid in the decision making for the UAV.

Management

Management contains all the actions or decisions the UAV may perform such as a request to perform entropy, move to waypoints, etc.

GUI

This has all the functions to produce the Graphical User Interface to select the necessary options for the UAV control.

Class Outline

CentralControl

Member Variables

strategy – Stores a reference to the current strategy

uavs - Stores a reference to all the UAVs in the system.

Vmax - Stores the max speed limit for the UAVs.

maxDistance - Stores the static max distance between UAVs.

minDistance - Stores the static min distance.

waypoints - Stores a 2-dimensional array of x and y positions for the swarm to target.

Num_of_uav – Saves the number of UAVs for the system.

Sep_distance – Saves the separating distance between the UAVs in the base unit of the environment.

Row_length – Saves the amount of UAVs on any given row of the UAV initial formation

Member Functions

initSysytem

- **Actions Performed:** Adds UAVs into the internal representation with initial data set.
- **Arguments:**
  - numberOfUAVs - Int - the number of UAVs in the system
  - separatingDistance - Float - distance between UAVs
  - numberOfRows - Int - how many rows to divide the formation of UAVs into.
- **Return Value:** None

addUAV

- **Actions Performed:** Instantiates a new UAV and adds it to its own collection of UAVs.
- **Arguments:**
  - ID - String - the name of the UAV
  - x - Float - initial x coordinate
  - y - Float - initial y coordinate
  - z - Float - initial z coordinate
- **Return Value:** None

Connect_to_environment

- **Actions Performed:** Establishes a connection to the environment based on the strategy.
- **Arguments:** None
- **Return Value:** Boolean – Crash occurs if connection could not be made and program terminates.

Close_connection

- **Actions Performed:** Disables the UAVs in the environment and terminates connection.
- **Arguments:** None
- **Return Value:** None

call_uav_update

- **Actions Performed:** Tells a given UAV to update its current X, Y, Z, and Orientation position.
- **Arguments:** None
- **Return Value:** None

uav_init

- **Actions Performed:** Gives all the UAVs in the system an initial take off/hover command.
- **Arguments:** None
- **Return Value:** None

entropyFormation

- **Actions Performed:** Commands UAVs to update their positions, calculate their entropy, determine their next move, and finally a movement command to AirSim. Also assigns waypoint targets for the UAVs.
- **Arguments:** None
- **Return Value:** None

write_data

- **Actions Performed:** Writes the UAV position and entropy to a file
- **Arguments:**
    - uavs - An array containing all the UAVs in the system
    - timeDiff - Float - The difference in time from the last data write
- **Return Value:** None

plotter

- **Actions Performed:** Creates a visual demonstration of the UAV positions and Entropy over the course of the simulation.
- **Arguments:** None

- **Return Value:** None

UAV

Member Variables

manager - Stores a reference to its Management object.

calculator - Stores a reference to its Calculations object.

Strategy – Stores a reference to its Strategy object.

ID - Stores the ID of the UAV.

intialX - Stores the initial x position of the UAV. This is used in the entropy formation.

intialY - Stores the initial y position of the UAV. This is used in the entropy formation.

intialZ - Stores the initial z position of the UAV. This is used in the entropy formation.

minDistance - Stores the UAV's designated minimum distance from other uavs.

maxDistance - Stores the UAV's designated maximum distance from other uavs.

maxVelocity - Stores the UAV's designated maximum velocity.

x - Stores the UAV's current x position.

y - Stores the UAV's current y position.

z - Stores the UAV's current z position.

theta - Stores the UAV's current theta orientation.

waypoint - Stores the UAV's current waypoint target.

Member Functions

identify

- **Actions Performed:** Returns the UAV's ID.
- **Arguments:** None
- **Return Value:** A formatted print statement of the UAV's ID.

getEntropy

- **Actions Performed:** Calls to get all the other UAV's positions then uses that to call the calculator to calculate its own entropy.
- **Arguments:** None
- **Return Value:** The UAV's own entropy.

getDistances

- **Actions Performed:** Calls to get the other UAV's positions then uses that to call the calculator to calculate this UAV's distance and angle to every other UAV.

- **Arguments:** None
- **Return Value:** A 2-dimensional array of distances and angles of this UAV to every other UAV.

Management

Member Variables

parent - Stores a reference to the parent UAV.

Member Functions

None

Calculations

Member Variables

parent - Stores a reference to the parent UAV.

Member Functions

calculateDistancesAndAngles

- **Actions Performed:** Uses the distance formula and arctan2 to calculate the distance and angle to each other UAV.
- **Arguments:**
  - otherUAVs - an array of references to each other UAV.
- **Return Value:** A 2-dimensional array of distances and angles to each other UAV.

calculateEntropy

- **Actions Performed:** Uses the Tsallis Entropy equation to determine what a given UAV's entropy is.
- **Arguments:**
  - otherUAVs - an array of references to each other UAV.
- **Return Value:** The entropy of the UAV.

calculateNextMove

- **Actions Performed:** Uses the given UAV's entropy and positions of the other UAVs to determine and calculate what the UAV should do next.
- **Arguments:**
  - otherUAVs - an array of references to each other UAV.
- **Return Value:** An array containing the UAV's desired Velocity and Orientation.

<Strategy>

Member Functions – ABC so no implementation

Establish_connection

Get_other_uav_positions

Get_self_position

Connect_to_environment

Close_connection

Move_by_heading

AirSimStrategy

Member Variables

Parent – Reference to the parent UAV

Master – Reference to the Central Controller

Default_json_save_location – String – Stores default location for the settings.json

Client – Reference to the AirSim Client

UNIQUE_SAVE_LOCATION – String – Custom location to store the settings.json

Member Functions

Establish_connection

- **Actions Performed:** Saves a reference to the central controller
- **Arguments:** Reference to the central controller
- **Return Value:** None

Get_other_uav_positions

- **Actions Performed:** Gets a list of all the UAVs in the system and returns a list of the UAVs with their position
- **Arguments:** None
- **Return Value:** None

Get_self_position

- **Actions Performed:** Gets the current status of the UAV's X, Y, Z and Theta orientation.

- **Arguments:** None
- **Return Value:** [X position, Y position, Z position, Theta Angle]

Connect_to_environment

- **Actions Performed:** Generate the AirSim Settings JSON and connect to the AirSim environment
- **Arguments:** None
- **Return Value:** Boolean – True if connection was made successfully

Close_connection

- **Actions Performed:** Disarm UAVs and close connection to them
- **Arguments:** None
- **Return Value:** None

Move_by_heading

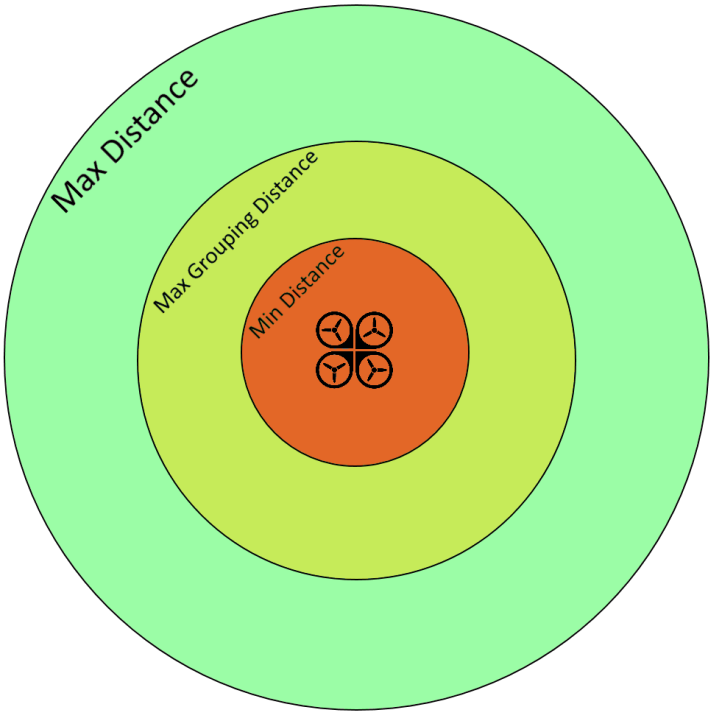
- **Actions Performed:** Call AirSim Client to move UAV
- **Arguments:**
  - X velocity - Float
  - Y velocity - Float
  - Z velocity - Float
  - Duration of move - Float
  - Angle in Radians - Float
- **Return Value:** None

Changes from Luke's Code

- Entropy Threshold will need to be made into an equation that takes in the number of UAVs
- UAVs now have a minimum grouping distance that each UAV tries to put all other UAVs in if their Entropy has not reached the threshold (See Figure 1)
- Entropy calculation now only uses the distance a given UAV is from each other UAV
- UAVs now check where a UAV is in relation to itself if it is too close (e.g. front left, front right, or behind) and turn accordingly (See Figure 2)
- UAV waypoint reassignment logic has changed
- UAV waypoint reached logic has changed

Figure 1

This Figure represents the different ranges that a single UAV will monitor for other UAVs.



- Within the Min Distance, no other UAVs should be this close to the UAV. Any UAVs that do enter this range (depending on where they are), the UAV will make measures to separate - see figure 2.
- Within the Max Grouping Distance, UAVs will attempt to place all the other UAVs in this range. If they are too close, they will separate, and if they are too far, they will move together.
- Within the Max Distance, the UAVs will measure their distances and attempt maneuvers to move closer. Anything outside this distance is not weighted more heavily but will still contributed to high entropy.

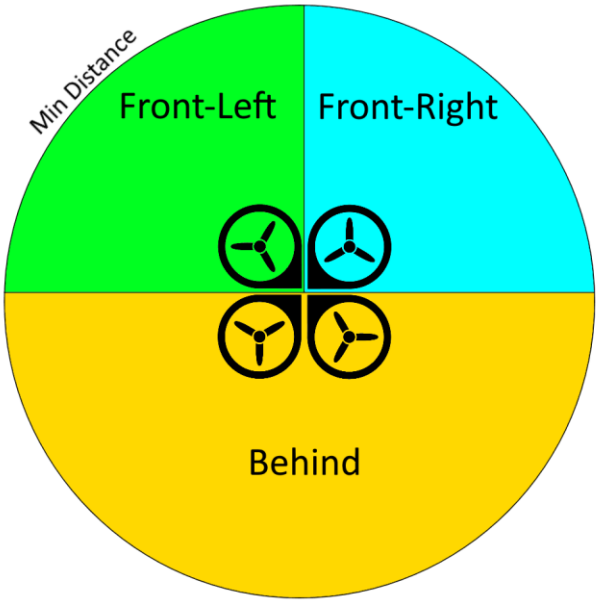


Figure 2

This figure represents the three sectors within the Min Distance that the UAVs will monitor. If another UAV exists in any of the sectors, the UAVs will attempt to separate but the sector will determine how they will separate. If another UAV is in the "Front-Left" sector, this UAV will determine the angle to that other UAV and subtract 90 degrees thus making it turn right and avoiding the other UAV. Similarly, in the "Front-Right" sector, the UAV will determine the angle to the other UAV and add 90 degrees to turn left. If a UAV is determined to be in the "Behind" sector, 180 degrees will be added (and readjusted to between 0 and 360) thus making the UAV move directly away from the other UAV