Inter IIT Tech Meet 11.0

PLUTO DRONE SWARM CHALLENGE

Task 2 code documentation

Submitted by - : Team 27

FEBRUARY 2023

Contents

1 Package Installation

The following packages are required to be installed for the codes to work (make sure python version is 3.6 or higher):-

- Python library manager

```
$ sudo apt-get update
$ sudo apt-get -y install python3-pip
```

- opencv-contrib version 4.7

```
$ pip install opencv-contrib-python==4.7.0.68
```

- numpy

```
$ pip install numpy
```

2 Class: *quadControl*

This class computes the pitch, roll and yaw commands based on the desired position using a PID Controller.

2.1 Methods

- __init__
  - Description: This is the constructor of the class. It performs the following actions:
    1. Creates an object of class *plutodrone* and establishes connection with the drone.
    2. Detects the ArCuo marker and sets the home position.
    3. Initialises the desired position and velocities.
    4. Initialises the gains of PID controller and the errors.
    5. Sets the upper and lower bounds on throttle, pitch, yaw and roll.
    6. Defines the frequency at which the packets are sent.
    7. Initialises the moving avergae filter.
  - Input: None
  - Output: None
- posControl
  - Description: This method computes the required roll, pitch, yaw and throttle values based on the desired position and velocities. The algorithm is as follows:
    1. Check the disarm flag, if its set to 1, then land the drone, disarm and disconnect.
    2. Update the current position of the drone using the ArCuo marker.
    3. Update the desired values from the parameters passed to the function.
    4. Calculate the errors used in the PID controller.
    5. Calculate the desired roll, pitch and throttle values.
    6. Limit the desired throttle, roll and pitch values to the lower and upper bounds.
    7. Send the command to the drone.
  - Input:
    * x : Denotes the desired x co-ordinate.
    * y : Denotes the desired y co-ordinate.

    - xvel : Denotes the desired velocity in x-direction.
    - yvel : Denotes the desired velocity in y-direction.
    - disarm : This is a flag used to land the drone in case of emergency.
  - Output: None
- getCurrentPose
  - Description: This method is used to get the pose of the drone from the camera using the ArCuo marker. The method performs the following actions:
    1. Populates the moving filter array with new values.
    2. Computes the current position of the drone.
    3. Computes the current velocity of the drone.
    4. Updates the current position of the drone.
  - Input: None
  - Output: None

3 Class: *hover*

This class is used to hover the drone at a desired height using ArUco marker.

3.1 Methods

- __init__
  - Description: This is the constructor of the class. It performs the following actions:
    1. Creates an object of class *quadControl*
    2. Initialises the ArCuo position as home position.
    3. Calculates the desired hover position.
    4. Sets the desired velocities.
    5. Stores the current time.
  - Input: None
  - Output: None
- path
  - Description: Calls the *posControl* function using the desired values.
  - Input: None
  - Output: None
- set_hover_time
  - Description: Repeatedly calls the *path* function till the hover time has not been reached. In case of emergency, the code can be interrupted using keyboard, which automatically lands the drone.
  - Input:
    - hover_time : The desired hover time
  - Output: None

4 Class: *move_rect*

This class moves the drone in a rectangle using ArUco marker.

4.1 Methods

- __init__
  - Description: This is the constructor of the class that initialises the drone position and also the desired position and velocities.
  - Input: None
  - Output: None
- rect
  - Description: This function generates the corner points of the rectangle and calls a function to move the drone to the desired position.
  - Input: None
  - Output: None
- path
  - Description: Calls the *posControl* function using the desired values.
  - Input: None
  - Output: None
- set_target_time
  - Description: Repeatedly calls the *path* function till the target time has not been reached. In case of emergency, the code can be interrupted using keyboard, which automatically lands the drone.
  - Input:
    - target_time : The desired time in which the drone should reach its goal position
  - Output: None

5 Class: *camera_pose*

This class detects the ArUco marker with the camera calibration parameters in place and gives the estimated position with respect to the camera centerd coordinate frame. The getPose() method is called by the quadControl class to take the position update at each iteration.

The pose estimation part is referenced from here