Getting Started

Installing the platform

1. Install ROS Kinetic according to the instructions - kinetic/Installation/Ubuntu
2. Install the relevant dependencies
   - Joy http://wiki.ros.org/joy
   - Gtkmm https://www.gtkmm.org/en/download.html
   - natnet_ros https://github.com/mje-nz/natnet_ros
   - vrpn_client_ros https://answers.ros.org/question/285887/ros_vrpn_client-installation-help/
3. A ROS workspace should be created using the following guide - http://wiki.ros.org/catkin/Tutorials/create_a_workspace
4. Navigate to *catkin_ws/src* and clone crazyflie_ros, then building the repository according to the instructions provided - whoenig/crazyflie_ros: ROS Driver for Bitcraze Crazyflie
5. Navigate to *catkin_ws/src/crazyflie_ros/crazyflie_controller/config/crazyflie2.yaml* and change the Z controller values to -
   a. kp: 30000.0
   b. kd: 15000.0
   c. ki: 200.0
6. The multi drone platform should first be cloned from the relevant source using the last stable build into your catkin_workspace. Using
   a. GitLab - https://gitlab.adelaide.edu.au/a1706141/mdp
   b. GitHub - https://github.com/jstollznow/multi_drone_platform
7. Following, the platform should be built by navigating to *catkin_ws* and executing the command - `catkin_make`

Repository structure

Each subdirectory will briefly be summarised to provide user clarity -

- documentation - Contains relevant platform information files, including Doxygen generated documentation. Doxygen generated documentation can be navigated by opening *catkin_ws/src/multi_drone_platform/documentation/doxygen/html/index.html*
- example - Contains a number of bash files to add a given number of virtual drones to the drone server and an example rviz (ros visualisation tool) configuration to visualise the drones on the drone server (up to 10, more can be added)
- include - Rigidbody and User API header files
- launch - Selection of ros launch files which can be used rather than launching each node individually.
- matlab - Contains all MATLAB related files including the API and relevant test scripts.
- msg - Contains two custom ros messages used internally.
- Results - Includes some results from thesis testing, specifically collision avoidance tests.
- scripts - Includes a number of programs and scripts testing and exemplifying platform features
- sessions - Includes dated session folders for each historic platform run (triggered through the use of the Live View windows).
- src - Contains a number of core features of the platform including
    - collision_management
        - artificial potential fields - drone-obstacle collision avoidance
        - static physical management - boundaries, error checking, physical limits etc.
    - debug
        - debug_app - Live View Window GUI
        - logger - Global logging procedure
    - drone_server - Drone management module
    - icp_implementation - Iterative closest point methodology
    - teleop_control - Allows remote control of each drone on the drone server through either a PS3 or PS4 remote
    - user_api - Contains the cpp API implementation
- srv - Contains the add_drone service to allow dynamic additions of drones while platform is live
- thirdparty - Contains some third party packages.
- wrappers - Includes the wrapper implementation files for each drone type. Currently has cflie, object, tello, vflie.

Order of execution

Ros nodes should be started in a specific order to ensure integration between different aspects is managed appropriately, the order is as follows -

1. Start roscore
2. Motion tracking
3. If using crazyflie drones, crazyflie server
4. MDP drone server
5. Add drones to drone server (can write a bash script to automate this process, some have been written in the *multi_drone_platform/examples* directory)
6. Launch debug windows for each drone (optional)
7. Launch MATLAB data generation (optional)
8. Launch script or program

Useful Commands

- Launch motion tracking
    - natnet - `rosrun natnet_ros _server:=129.127.29.166`
    - vrpn - `roslaunch vrpn_client_ros sample.launch server:=129.127.29.166`
- Launching crazyflie server - `rosrun crazyflie_driver crazyflie_server`
- Drone server - `rosrun multi_drone_platform drone_server`
- Adding drones to your drone server
    - Via command line arguments - The drones name (motion capture tag) followed by the specific arguments for the drone type associated with the tag. Examples of adding each of the two main drone types can be seen below -
        - vflie - `rosrun multi_drone_platform add_drone <tag> <homePosX> <homePosY>`
            - `rosrun multi_drone_platform add_drone vflie_00 0.50 1.00`
        - cflie - `rosrun multi_drone_platform add_drone <tag> <linkURI> <droneAddress>`
            - `rosrun multi_drone_platform add_drone cflie_E7 radio://0/80/2M E7`
    - Via prompts - same command as above but with no arguments, prompts will guide the adding process, this is good to learn to arguments associated with each drone type
- Launch debug windows
    - Expanded - `rosrun multi_drone_platform all_debug_windows expanded`
    - Compressed - `rosrun multi_drone_platform all_debug_windows compressed`
- Launch MATLAB data recording - running the bash script called *matlab_data.sh* in the examples local directory. This bash file may need to be altered depending on the install location of MATLAB.
- Each script can be run using the same basic template - `rosrun multi_drone_platform <script_name>`
    - yaw_control - `rosrun multi_drone_platform yaw_control`

Useful follow ups

- Contained within this documentation directory is a number of other information files which detail a number of platform components including -
    - Rigidbody object
    - User API
    - Live View Window GUI
    - Teleop Control
    - How to create a custom drone wrapper