D Rigidbody

D.1 ROS topics

Each rigidbody object frequently updates the following ROS topics (assuming the selected rigidbody has numeric ID of 0) –

- **/mdp/drone_0/apiUpdate:** The drone server uses each relevant rigidbody object to publish API requests to this topic. Internally, as clarified in Section 2.6.4, requests published on this topic are then asynchronously checked for validity before being added to the rigidbody's internal command queue. This ROS topic should not be manipulated externally.
- **/mdp/drone_0/battery:** This topic publishes the battery level of the drone if it is accessible through the selected drone ROS driver. The update frequency is determined by the ROS driver and the developer.
- **/mdp/drone_0/closest_obstacle:** This topic changes every update cycle as called from the drone server. This topic publishes the distance to the closest object at each iteration. This object could be another drone or a tracked obstacle.
- **/mdp/drone_0/obstacles:** This topic changes every update cycle as called from the drone server. This topic publishes an array of obstacle positions relative to the subject drone, sorted by distance in ascending order. The message is of the type *geometry_msgs/PoseArray*.
- **/mdp/drone_0/curr_pose:** This topic is updated according to each motion capture update. The topic publishes the current position of the drone in the

form of a *geometry_msg/PoseStamped*. Each message includes the linear and angular orientation of the drone with a ROS timestamp.

- **/mdp/drone_0/curr_twist:** Similar to */curr_pose*, this topic publishes the drones velocity according to the last two positions provided from external motion capture topics. Each message also has a timestamp and is of the type *geometry_msgs/TwistStamped*.
- **/mdp/drone_0/des_pose:** The rigidbody object publishes a *geometry_msg/PoseStamped* message to this topic when the desired position of the drone is altered either internally or as requested through the API. The timestamp is associated with the ROS timestamp at which the desired position was altered.
- **/mdp/drone_0/des_twist:** The rigidbody object publishes a *geometry_msg/TwistStamped* message to this topic when the desired velocity of the drone is altered either internally or as requested through the API. As with the desired position, the timestamp reflects the time at which the desired velocity was altered.
- **/mdp/drone_0/log:** As mentioned in 2.6.3 each drone maintains a log topic in which all output messages seen in the relevant terminal window are also published as messages to this topic.

D.2 Wrapper abstract methods

Each wrapper class is required to inherit the rigidbody object and implement the following methods, for brevity all function parameters will be ignored, it should be noted all abstract functions do not return any value –

- **on_init:** This function is called following the initialisation of the rigidbody. The rigidbody object is created whenever an object is added to the multi-drone platform. It is recommended any related ROS initialisation required for specific drone should be initialised in this method.

- **on_deinit:** This function should appropriately dispose of any objects before the object is deleted and the object reference is lost. The user should terminate the communication protocol connection in this function. This function is called whenever a global shutdown procedure is called or the given rigidbody is removed.
- **on_update:** The on update method is called from the rigidbody update function and thus is called each drone server iteration. In the case of the *vflie* this function is used to generate the next position based on its current commands and linear interpolation principles as outlined in Section 2.6.3.
- **on_motion_capture:** As the name implies this function is called when a new position message is posted on the motion tracking topic following the update of current position and current velocity. If the ROS driver used to send drone commands requires position updates, this function could be used to forward the last known position of the drone to the relevant ROS topic or node. This function is used for that exact purpose in the *cflie* wrapper class.
- **on_takeoff:** Called when a takeoff command is requested, this function should communicate that request to the drone through the drone type's preferred communication protocol.
- **on_land:** Called when a land command is requested. This request should be passed on to the drone.
- **on_emergency:** Called when a emergency command is requested. This function should immediately power down the drone as quickly as possible.
- **on_set_position:** Called when the desired position of the drone is set. This should command the drone to the goal position in the specified duration. If the drone's ROS driver does not allow for position control, or the drone itself does not allow for position control the position and duration should be converted into a velocity command.
- **on_set_velocity:** Called when the desired velocity of the drone is changed.

This should set the velocity of the drone for the requested duration. If the drone does not allow for velocity control, the velocity should be mapped to a position control command.